NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PREDICTING HOST LEVEL REACHABILITY VIA STATIC ANALYSIS OF ROUTING PROTOCOL CONFIGURATION

by

Stephen McManus, Jr.

September 2007

Thesis Advisor:	Geoffrey Xie
Second Reader:	J. D. Fulp

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Predicting Host Level Reachability via Static Analysis of Routing Protocol Configuration		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Stephen McManus, Jr.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Static analysis refers to techniques that extract and check the semantics of a program *entirely* from examining its source code. In this case, router configuration files can be thought of as the source code of a distributed program whose execution determines the host level reachability of the network. Static analysis brings about new challenges. Unlike a regular computer program, router configuration commands hide the detailed logic of routing protocols. Completely constructing the logic for static analysis of router configuration files is difficult and even impossible in some cases where the network has a large number of concurrently running routing processes distributed over many routers and variable network delays make the interactions between these processes too complex to understand exactly.

This research takes an initial step in understanding the power of static analysis. A system is built to infer the packet forwarding table of each router from the router configuration files. The scope of the work is confined to networks where OSPF is used exclusively for routing. The system is able to infer the exact forwarding tables of the Cisco routers for several lab test networks.

14. SUBJECT TERMS OSPF, Static Analysis, Packet Forwarding Table, Forwarding Information Base, Router Redistribution, Route Selection			**15. NUMBER OF PAGES** 167
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PREDICTING HOST LEVEL REACHABILITY VIA STATIC ANALYSIS OF ROUTING PROTOCOL CONFIGURATION

Stephen C. McManus, Jr.- Civilian, Federal Cyber Corps
B.S. in Computer Science, Saint Louis University Parks College, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2007

Author: Stephen McManus, Jr.

Approved by: Geoffrey Xie
Thesis Advisor

J. D. Fulp
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Static analysis refers to techniques that extract and check the semantics of a program *entirely* from examining its source code. In this case, router configuration files can be thought of as the source code of a distributed program whose execution determines the host level reachability of the network. Static analysis brings about new challenges. Unlike a regular computer program, router configuration commands hide the detailed logic of routing protocols. Completely constructing the logic for static analysis of router configuration files is difficult and even impossible in some cases where the network has a large number of concurrently running routing processes distributed over many routers and variable network delays make the interactions between these processes too complex to understand exactly.

This research takes an initial step in understanding the power of static analysis. A system is built to infer the packet forwarding table of each router from the router configuration files. The scope of the work is confined to networks where OSPF is used exclusively for routing. The system is able to infer the exact forwarding tables of the Cisco routers for several lab test networks.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. DUE-0414102. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to thank Nicole McManus and Marcin Pohl for editing my paper. JD Fulp has been a great help as a second reader. Dr. Geoffrey Xie has helped with the understanding of networking topics and the process for writing thesis. I would like to thank my friends and family for their support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Host level reachability of a network refers to the exact types of packets that may currently traverse the network between each pair of end hosts in the network. Today, when a host level reachability problem occurs, e.g., a FTP server is not responding to requests from a particular machine, the network administrator typically uses "ping" or "traceroute" to troubleshoot. These kinds of probing techniques have severe limitations. First, they only check the reachability of certain types of ICMP packets and the results may not apply to the problem at hand. Second, they require access to the live network and thus cannot be used for "what if" analysis without costly disruptive testing [1].

Recently, an alternative approach which models host level reachability from a static analysis of router configuration files has been proposed to address these two problems [1]. Static analysis refers to techniques that extract and check the semantics of a program *entirely* from examining its source code. In this case, router configuration files can be thought of as the source code of a distributed program whose execution determines the host level reachability of the network. Static analysis brings about new challenges. Unlike a regular computer program, router configuration commands hide the detailed logic of routing protocols. Completely constructing the logic is difficult and even impossible in some cases where the network has a large number of concurrently running routing processes distributed over many routers and variable network delays make the interactions between these processes too complex to understand exactly. Also, routing protocols are not the only factors in determining host level reachability. The content of the packet forwarding table at each router, formally known as the Forwarding Information Base (FIB), is one of the key factors that impact host-level reachability. Other mechanisms such as packet filters and Network Address Translation (NAT) devices are installed in many networks [1].

A. OBJECTIVE

This research explores the static analysis of router configuration files. The scope of the work is confined to developing a method for predicting host level reachability via a

static analysis of those router commands for configuring routing protocols. This thesis infers the packet forwarding table at each router from router configuration files. Static analysis of Cisco routing configuration files use a white-box approach by capturing router configuration files and producing FIB tables offline.

B. BENEFITS OF STATIC ANALYSIS

Static analysis has several benefits. First, the network will suffer no disruption from this analysis. Static analysis is done completely offline, and as such does not cause breaks in service. Second, "what if" [1] analysis can be performed before the construction of a network.

C. RESEARCH QUESTIONS

a. What is a suitable abstraction for modeling the flow of routing information between routers that we can derive from parsing router configuration files?

b. How accurately can we estimate the FIB contents of routers based on this route flow abstraction?

c. What is a good method for synthesizing and presenting network wide reachability results from the estimated FIB contents?

d. Can Java be used to implement a static reachability analysis tool based on this research?

e. How can we make the static analysis tool extensible so that it will be easy to incorporate future enhancements to both the route flow abstraction and the FIB estimation algorithm?

D. ORGANIZATION

Chapter II provides a background understanding of routing protocols and the algorithm [2] used for route redistribution. Chapter III describes the steps taken to solve the route redistribution problem. Chapter IV describes a Java implementation of an

actual system. Chapter V describes the validation of the system using three sets of network configurations. Lastly, Chapter VI concludes the research and makes recommendations for the future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The purpose of the thesis is to build a system for estimating packet forwarding tables from router configuration files within a network. A router's packet forwarding table, commonly referred to as the FIB, provides the next hop information (i.e., a route) for a given a destination IP address. Route selection and route redistribution are the two main processes that routers perform to determine which routes to insert into the router's FIBs. The first part of this chapter, including Sections A and B, describes the route selection and route redistribution processes. This thesis focuses on networks that utilize the OSPF routing protocol. Therefore, an overview of OSPF is provided in the third part of this chapter.

A. ROUTE SELECTION

A router that runs multiple routing protocols actually instantiates a separate *routing process* for each protocol. Every instantiated routing process has its own Routing Information Base (RIB) to store protocol-specific routing information [2]. These routing processes may derive different routes to the same destination prefix from their respective RIBs and select one of the routes to install into its FIB for that destination. When more than one routing process have routes to a network prefix, a routing process configurable parameter called administrative distance is used to decide which route is installed into the FIB. All routing processes have a default administrative distance, such as 110 for OSPF. Setting administrative distances can be useful to gain flexibility within a network. In Figure 1, Router 1 installs routes into the router's FIB from three RIBs. Routing process OSPF 1 contains all routes within OSPF area 1. So, routing process OSPF 2 contains all routes within OSPF area 2. The FIB table for Router 1 contains routes to network prefixes either from routing process OSPF 1, Static or OSPF 2. For example, routing process OSPF 1 knows of a route to a network prefix of 10.1.12.12/24 through interface Ethernet1/0. This route will be installed into Router 1 FIB table. Also, routing processes may have routes to common network prefixes.

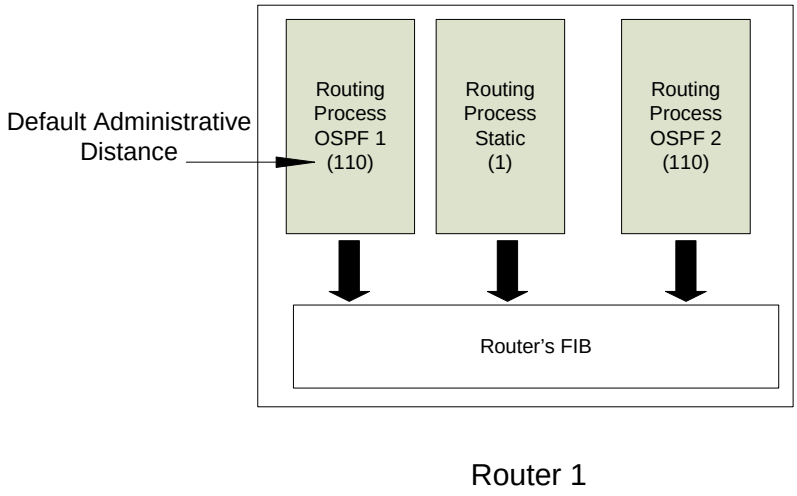

Figure 1. The creation of the FIB table for Router 1.

B. ROUTE REDISTRIBUTION

Route redistribution is a mechanism introduced by router vendors to facilitate dissemination of routes from one routing process into another running on the same router, e.g., from OSPF 1 to OSPF 2 in Figure 1. This thesis considers three types of routing processes in particular. They are: OSPF and the two built-in routing processes for handling static routes and connected subnets. Route redistribution (RR) can be configured between similar routing instances (i.e., multiple OSPF areas) or completely different types of routing instances using distinct routing protocols. If improperly configured, RR can cause problems, such as routing loops, stemming from the interaction between different routing instances. In Figure 2, Router 2 is configured to redistribute routing information from RIP to OSPF and from OSPF to RIP. In this scenario, Router 2 will collate routes from the RIP network and advertise the availability of network prefixes known from Router 1 to the OSPF network. Conversely, Router 2 will collate routes from the OSPF network and advertise the availability of network prefixes known from Router 3 to the RIP network. Once all of the sharing of information has been completed, Router 1 will know all the network prefixes available on Router 3 and Router 3 will know all the network prefixes available on Router 1. RR offers the ability to connect networks of different routing protocols together without requiring all of the routers to run the same routing protocol. A more detailed tutorial of RR can be found in [2].

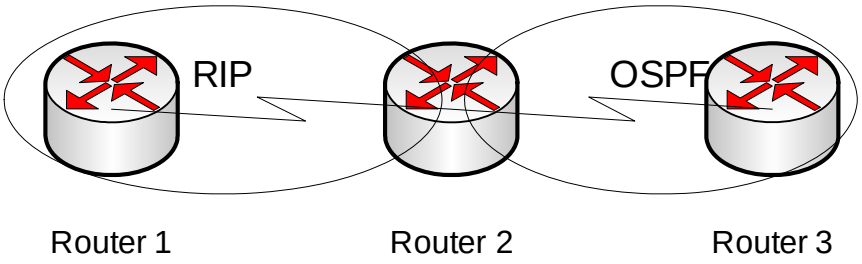

Figure 2. Route redistribution between two routing instances (RIP and OSPF).

C. OPEN SHORTEST PATH FIRST (OSPF) PROTOCOL

OSPF is a link state routing protocol that runs directly on top of the IP protocol [3]. The original design of OSPF was as an interior gateway protocol (IGP), but it can be extended for other uses [3]. OSPF is primarily an IGP because OSPF is typically used for exchanging routes between computers within a single network domain. The network operators have expanded the nature of OSPF into including complex interactions with multiple routers in multiple areas. Due to the increasing complexity in redistribution of information between routers, many problems have arisen with OSPF. Routing oscillations and "route flapping" are possible consequences of the vast number of possibilities with OSPF [2].

1. Single Area

OSPF has the ability to control single or multiple areas of routers. Let us first look at how OSPF operates when all routers belong to a single area. In Figure 3, all four routers are configured to be in the same OSPF area. Since all of the routers are in the same area, each router within an OSPF area will have a global view of the network. OSPF is a dynamic routing protocol that uses link state advertisements (LSA) and link state databases (LSD). A router uses LSA packets to broadcast all of the routes, either local or imported from other routing processes, and broadcasts this information to the rest of the routers within the same area. The format of the LSA packet [3] is displayed in Figure 4. Eventually, all routers within the same area contain the same LSD content. Figure 5 is an example LSD dump taken from a router in a network similar to that of Figure 3.

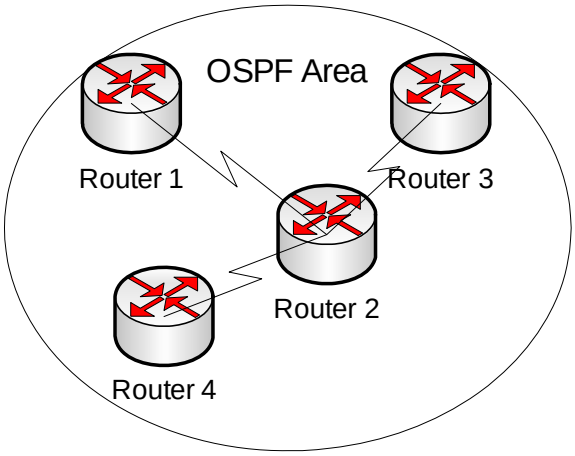

Figure 3. Routers within an OSPF area.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age            |     Options   |       1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Link State ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Advertising Router                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LS sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          LS checksum          |              length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    0    |V|E|B|         0       |            # links            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link Data                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     # TOS     |         TOS 0 metric          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TOS      |         0       |            metric             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TOS      |         0       |            metric             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link Data                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
```

Figure 4. LSA packet.

```
            OSPF Router with ID (10.2.14.14) (Process ID 11)

                Router Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum Link count
10.2.14.14      10.2.14.14      101         0x80000002 0x0085F4 1
10.2.24.24      10.2.24.24      107         0x80000002 0x00A38F 1
10.10.10.3      10.10.10.3      102         0x80000004 0x00BC1F 3
10.10.10.5      10.10.10.5      139         0x80000002 0x00B267 1

                Net Link States (Area 1)

Link ID         ADV Router      Age         Seq#       Checksum
10.1.13.31      10.10.10.3      102         0x80000001 0x005840
10.1.23.32      10.10.10.3      107         0x80000001 0x00EE8A
10.1.35.53      10.10.10.5      140         0x80000001 0x00224D

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
10.5.5.2        10.10.10.5      224         0x80000001 0x0064F3 0
10.10.10.3      10.10.10.5      222         0x80000001 0x007A77 0
10.10.10.5      10.10.10.3      145         0x80000001 0x00512B 0
10.10.10.5      10.10.10.5      233         0x80000001 0x00D278 0
192.168.1.0     10.10.10.5      231         0x80000001 0x0016E3 0
```

Figure 5. Link state database.

Once a router has received LSAs from all other routers within the OSPF area, the router will start to compute the shortest paths to the network prefixes found in the LSD. Each router uses an identical Dijkstra algorithm to compute the shortest internal routes to network prefixes avoiding the complications of loops within an OSPF area. Note that the link costs in OSPF are asymmetric. If multiple paths from a source router have the same cost to a destination network prefix, OSPF will cause a load balancing effect over the multiple paths [3].

2. OSPF Multiple Area

OSPF with multiple areas is a complex problem within networking. When multiple areas are represented within a network, areas cannot notify other areas of possible networks without using RR. In Figure 6, there are two separate instances of OSPF routing protocols. The routers within the OSPF area 2 network can send LSA packets to each other. Also, routers within the OSPF area 1 are allowed to send LSA

information between them. The problem appears when the routers from the OSPF area 2 network have no connectivity to the routers in the OSPF area 1 network.

OSPF routers are allowed to send LSA information between OSPF areas using redistribution. External redistribution is used to forward information from routing processes into an OSPF area. For different types of routing processes to communicate, the additional OSPF header is appended to the LSA packet advertising the external network prefix. All the routers in Figure 7 could use external redistribution to announce network prefix availability to other routers. Type 1 external redistribution has the same magnitude of OSPF cost as internal OSPF routes but type 2 external redistribution has a higher order of magnitude of cost than internal OSPF routes [3]. Therefore, type 1 external redistribution routes are normally preferred routes over type 2 external redistribution routes.

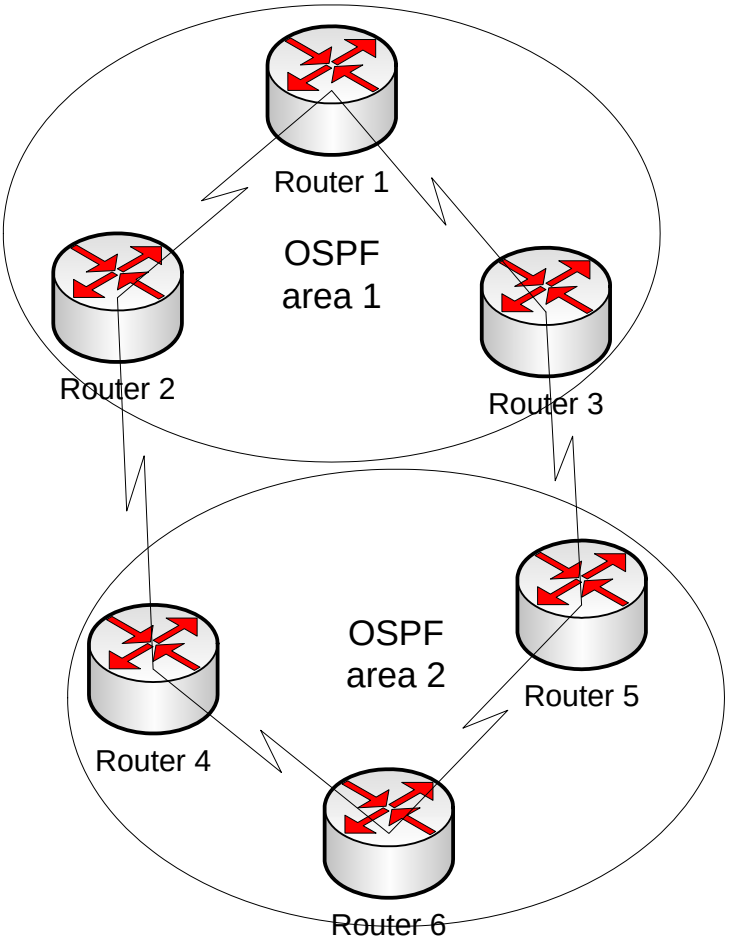

Figure 6. Redistribution between OSPF area 1 and 2.

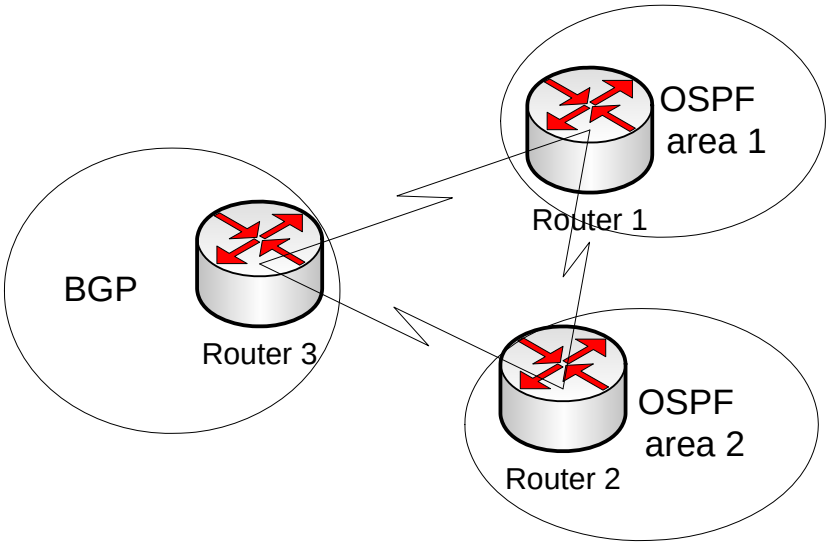

Figure 7. External redistribution example.

3. Route Redistribution between Two OSPF Areas

Most routing protocols will prefer internal routes over external routes, which is called convexity [2]. OSPF routers can be organized in such way to prefer external routes over internal routes by specifically setting the administrative distance. Administrative distances can be used to expose possible routing loops in certain routing protocols, e.g. OSPF. Administrative distances are used to state which possible path to a network prefix is more reliable than the other paths to the same network prefix. So, the administrative distance has a higher precedence in determining routing paths than the cost of the link. A route to a network prefix may contain the lowest cost through one particular path. If the administrative distance is higher for a particular link, OSPF will utilize a different link with a lower administrative distance even if the cost for that path is elevated. Figure 8 shows a network setup with a redistribution of two OSPF areas and administrative distances labeled as AD. As shown in Figure 9, all of the external OSPF paths from Router 1 are type 2 external redistributed, “O E2”. This happens due to the OSPF paths in OSPF area 1 having a higher administrative distance, 200, than the paths in OSPF area 2—administrative distance 100. Even though the path for network prefixes in OSPF area 1 would have a shorter cost going through Router 3, Router 1 will choose to

go through Router 4 because the link from Router 1 to Router 4 is specified to be more reliable than the link from Router 1 to Router 3.

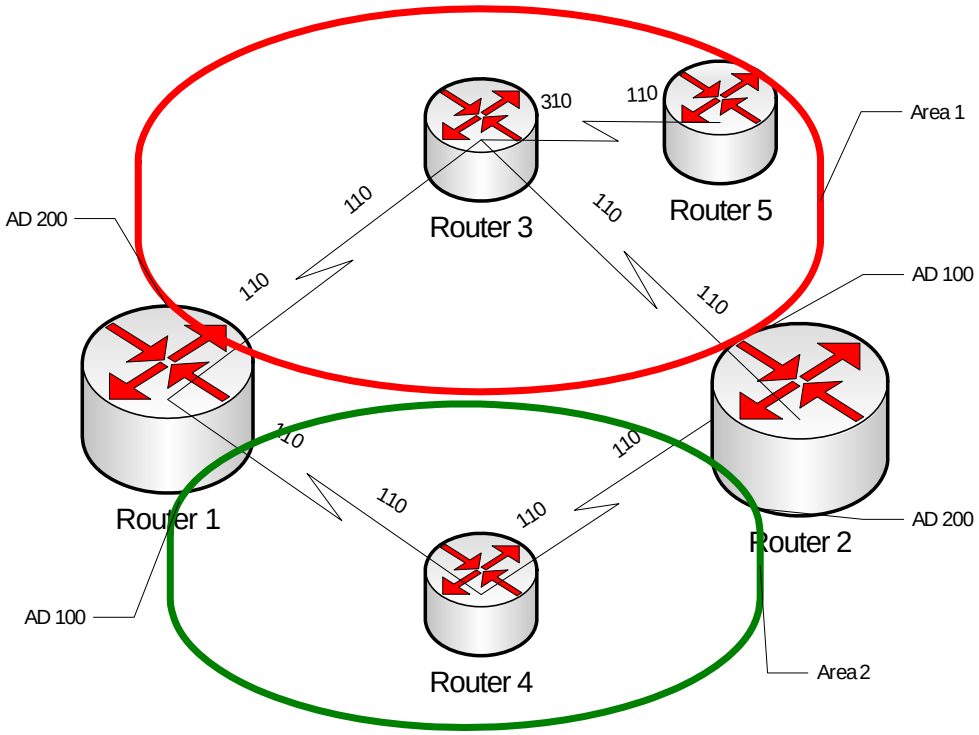

Figure 8. Target network composed of two OSPF areas.

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C       10.2.14.0/24 is directly connected, Ethernet1/3
C       10.1.13.0/24 is directly connected, Ethernet1/1
O E2    10.5.5.2/32 [100/100] via 10.2.14.41, 00:00:46, Ethernet1/3
O E2    10.10.10.3/32 [100/100] via 10.2.14.41, 00:00:46, Ethernet1/3
C       10.10.10.1/32 is directly connected, Loopback0
O E2    10.10.10.5/32 [100/100] via 10.2.14.41, 00:00:45, Ethernet1/3
O       10.2.24.0/24 [110/130] via 10.2.14.41, 00:00:46, Ethernet1/3
O E2    10.1.23.0/24 [100/10] via 10.2.14.41, 00:00:47, Ethernet1/3
O E2    10.1.35.0/24 [100/320] via 10.2.14.41, 00:00:47, Ethernet1/3
     192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [100/100] via 10.2.14.41, 00:00:47, Ethernet1/3

Router 2
```

Figure 9. Router 1 from the experiment.

D. MODELING ROUTE REDISTRIBUTION

The following is an overview of a model developed by Le, Xie, and Zhang for analyzing the routing dynamics of route redistribution [2]. The algorithm breaks the logic of RR down into 3 steps as described below. The example network shown in Figure 8 is used to illustrate each of these steps. The first phase is to abstract the route selection process of the border routers that connect multiple OSPF areas. The next phase is to produce a router redistribution graph that represents a comprehensive view of all the routing instances within the network. The last phase is called a "network-wide RR logic" which discovers the paths to network prefixes through multiple routing instances [2].

1. Per-Router Route Selection and Route Redistribution Logic

Since we are looking primarily between OSPF redistribution routes, the first step is to use Dijkstra's algorithm to install the best route to all network prefixes into the router's FIB.

> The installed route is called an active route and is redistributed to all of the connected routing instances. The algorithm is described in Figure 10 where P is the network prefix that the router r is trying to reach. V is the set of routing processes in r. U is an individual routing process on r that has a RIB and an administrative distance, which is u.RIB and u.ad. Selected-process(P) is the selected routing process to reach P and active-route(P) is the routing process inserted into the router's FIB to contact P. [2]

Procedure *Route selection at router r*

1: **for all** routing process $u \in V$ such that $P \in u.RIB$ **do**
2: **if** ($u.ad <$ *selected-process(P).ad*) OR ($u.ad ==$ *selected-process(P).ad* AND rand(0, 1) == 1) **then**
3: *selected-process(P)* $\leftarrow u$
4: *selected-process(P).ad* $\leftarrow u.ad$
5: Select best route from selected-process(P) (according to metric) and install it in FIB
6: *active-route(P)* $\leftarrow$ selected best route
7: **end if**
8: **end for**

Procedure *RR at router r*

1: **for all** routing process $u \in V$ such that redistribution of P from *selected-process(P)* to u is enabled **do**
2: Redistribute *active-route(P)* into u according to applied routing policies
3: **end for**

Figure 10. Route selection and route redistribution for each router from [2].

2. Route Propagation Graph

Then, the route propagation graph is a graph with nodes represented as the OSPF routing instances that are redistributing information and the edges are the routers that do the OSPF redistribution. The route propagation graph, in Figure 11, correlates to the network setup of Figure 8. The hashed vertex represents the originating routing process of the network prefix P. Several routing processes may identify a path to a destination network prefix.

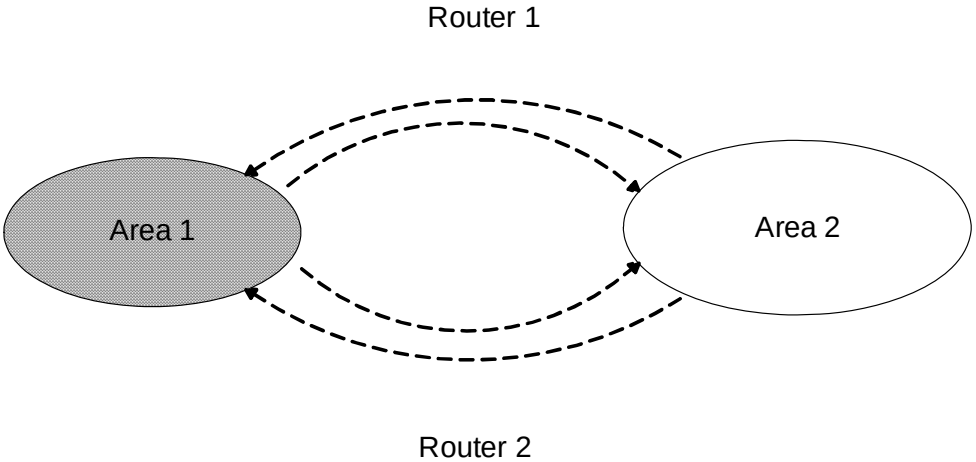

Figure 11. Route propagation graph from the experiment.

3. Network-Wide Route Redistribution Logic

With all of this information already collected, route redistribution is performed at a network-wide level. The algorithm is displayed in Figure 12 where CL is the list of routers that have an area waiting to be activated and S is the subset of routers that have been activated at each time interval [2]. At each time interval, a subset of the routers from CL are activated. When a routing instance is triggered by the activated routers, the routing instance performs route selection and route redistribution. Since the set of activated routers can be selected differently at each time interval, the output may be populated differently each implementation of the network-wide route redistribution logic.

Procedure *Network-wide RR*

Initialization

- $t = 0$
- We insert the redistributing routers that are connected to (i.e., have an edge either from or to) an originating vertex into CL.

Main loop

1: **while** $CL \neq EMPTY\ SET$ **do**
2: t++
3: Remove a subset S of routers from CL
4: **for all** router $r \in S$ **concurrently do**
5: Execute *route-selection logic* at r:
Among all vertices v that are connected to r and that have an eligible route (i.e., v is colored dark and have a router other than r redistributing a route into v, or v is striped), select the routing instance with the lowest ad.
6: **if** r's selected routing process has changed **then**
7: //let $r.u$ denotes the new selected routing process
8: Execute *route-redistribution logic* at r:
Each edge from u through r is activated while all other edges through r are deactivated
9: **for all** vertice v to which $<u,v,r>$ is active **do**
10: insert routers that have an edge from or to v into CL (if not already present)
11: **end for**
12: **end if**
13: **end for**
14: **for all** vertice $u \in V \setminus O$ **do**
15: **if** there is no active edge pointing into u **then**
16: Color u in white
17: **else if** there is an active edge pointing into u **then**
18: Color u in dark
19: **end if**
20: **end for**
21: **end while**

Figure 12. Network route redistribution algorithm from [2].

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

A system for estimating network wide FIB tables from router configuration files must implement two functions: parsing and static analysis, as shown in Figure 13. The first part of the system parses all router configuration files in the network, then extracts and stores all those commands related to reachability control. This thesis focuses on interface and OSPF related configuration commands. The second aspect of the system is to generate a FIB table for each router from the configuration parameters gathered. At the start of the thesis, a decision was made to use a database, instead of a data structure at run time, for storing parsed router configuration information. The rationale for this design decision will be discussed in Section A.

Figure 13 below provides a very high level illustration of the system concept, particularly the order of system operations. The system uses the database not only as the source of information, but also updates the database with intermediate data. The reason for having a double arrow between the static analysis module and the database is because static analysis will both read information from the database and store the intermediate results back into the database. As the last step, the sytem outputs the estimated router FIB tables from the final results stored in the database.

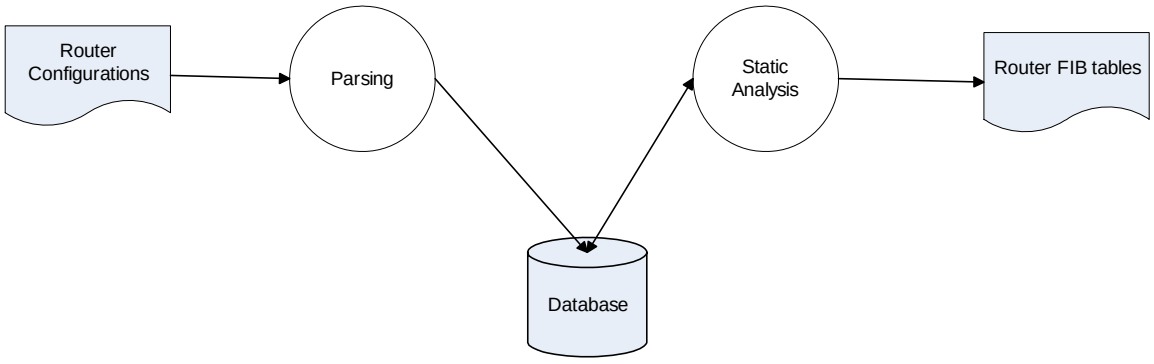

Figure 13. System concept.

A. DATABASE

The target system will require the correlation of configuration parameters from multiple routing configuration files. This can be easily achieved with a database and the Structured Query Language (SQL). Another appealing aspect of using a database is the separation of code from data. The data can be stored on a machine and the code can be executed from a different machine. If the system evolves from the original design, the data will remain the same in the database. Storing the data in a database also permits grouping of information into multiple tables reflecting the logical structure of gathered data. Also, a database seemed to be a good solution to deal with frequently altering data structures that need to be flexibly queried for further insight.

Built-in networking functionality in databases is useful in analysis that deals with IP addresses. Once the information is inserted in the table in a format for networking, there are several different representations that could be requested from the database. For example, the program can query for the network IP address and number of netmask bits used. Intersection and except clauses in some SQL statements are useful for some of the information. For example, without using the except-clause to return the routers that are in the same OSPF area excluding the current router, the program would have to iterate through the returned set and only add routers that are not the current router. A self-contained program could perform the previous logic sequence but using a database is more efficient.

Figure 14 shows the common architecture for a relational database and tables related to the database. Multiple tables can be populated in a database to hold information. For example, one of the tables could hold the information for each interface on each router. There is no single table that contains all the information from the router configuration files, as that would not follow the BNF standard form. Joining the multiple tables of information will give you all of the information gathered from the router configuration files.

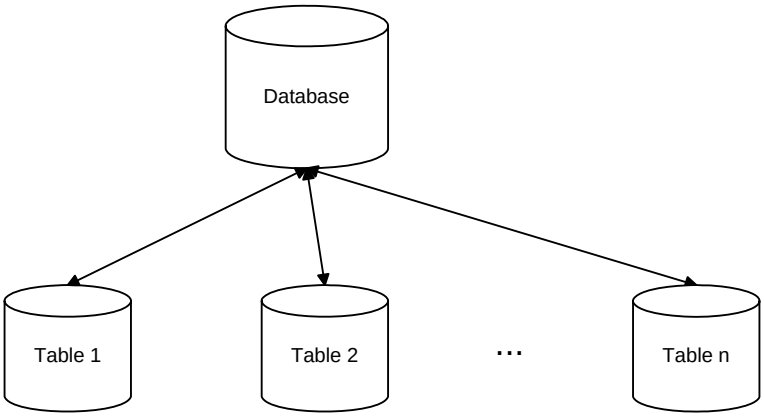

Figure 14. Database architecture.

B. PARSING

The first step for parsing the router configuration files is to collect them. The router configuration files are collected using the "show running" command from each router and collated into one text file. The parser linearly scans through the file, groups the information into four categories: interface, static, OSPF and redistributed information, and stores the information in the database. Other information like the IP addresses and OSPF link costs must also be collected. Figure 15 shows the flow diagram of parsing router configuration files.

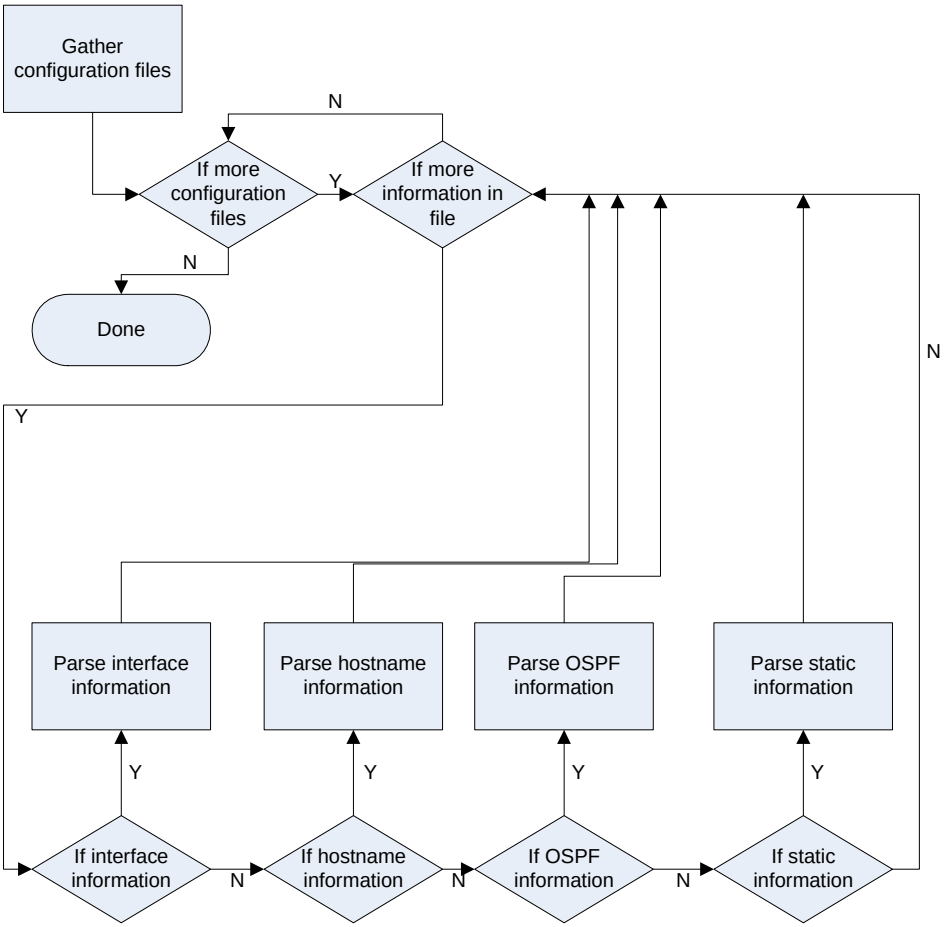

Figure 15. Flow chart of parsing router configuration files.

C. STATIC ANALYSIS

The static analysis functionality is handled by using multiple procedures. Figure 16 displays the logic sequence for analyzing OSPF and RR. The sequence reflects the timing flow of routing information in the network. The first step is to obtain the locally connected subnets for each router-pair within the network. The next step is to gather the static routes defined by each router. The next step is to enter the static and connected routes that are redistributed into an OSPF area into the OSPF RIB of every router within the same OSPF area. Then, the routes that are redistributed between OSPF areas are analyzed by the RR logic described in Chapter II. Finally, the FIB output is generated from the results of the prior functions.

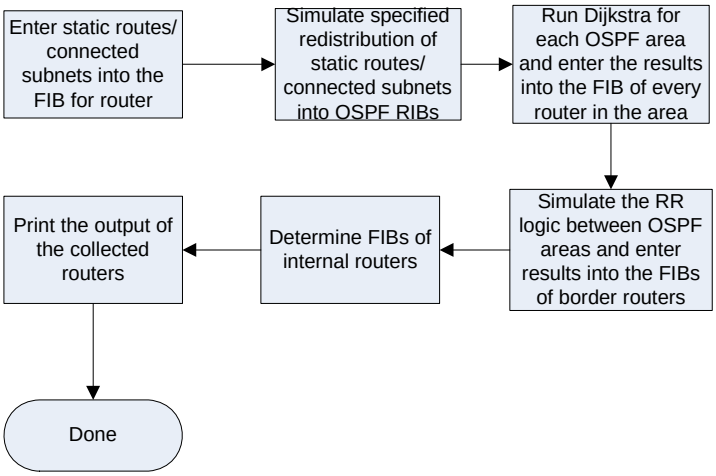

Figure 16. Flow chart of analyzing router configuration files.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION

Following the design described in the previous chapter, a prototype system has been constructed with a relational database backend and a Java frontend. The database part of the implementation is based on the open source PostgreSQL 8.1.5 package. All Java code has been compiled with the Java compiler version 1.5.0 from Sun Mircosystems. The Java code used for computing Dijkstra's algorithm is borrowed from [5,6,7].

A. DATABASE SCHEMA

List of relations

Schema	Name	Type	Owner
public	algorithm	table	mcmanst
public	instancetab	view	mcmanst
public	interface	table	mcmanst
public	mainospf	view	mcmanst
public	morered	table	mcmanst
public	ospf	table	mcmanst
public	output	table	mcmanst
public	redistribute	table	mcmanst
public	ritable	table	mcmanst
public	static	table	mcmanst
public	tempospf	table	mcmanst

Table 1. List of tables in the database.

Table 1 shows all the database tables and views that are defined within the system. Each of the tables has a specific purpose. The code will use some of the tables for various functions to eventually conclude with the universal FIB table, which is stored in the "output" table. The "interface" table includes the information from the router configuration files that correlates to every active interface on each router that has an IP address and is not "shut down". Knowing the interfaces for routers is necessary in determining connections between routers. The "interface" table includes the router's name, the hostname of the router, the interface identifier and the IP address of the interface.

Table
"public.interface"

Column	Type
routername	text
hostname	text
interfacename	text
ip	inet

Table 2. Interface specification.

The “static” table contains the static routes gathered by parsing the router configuration files. Static routes are an integral part to routing and are used often when redistributed within an OSPF area. The “static” table contains the router’s name, the IP address of the destination network prefix and IP address of the interface used for the static route.

Table
"public.static"

Column	Type
routername	text
ip	inet
interfaceip	inet

Table 3. Static specification.

The “redistribute” table contains all routes that will be redistributed into an OSPF area or between multiple OSPF areas. The types of redistribution analyzed in this system are connected, static and OSPF. The previous information is gathered from the router configuration files for the system to compute OSPF routes that have been redistributed into an OSPF area. The “redistribute” table contains the router’s name, OSPF label to enter the redistributed route, the type of redistribution, an OSPF label of routes to redistribute from and the cost of the redistribution.

Table
"public.redistribute"

Column	Type
routername	text
ospfinput	text
protocol	text
ospfoutput	text
cost	integer

Table 4. Redistribute specification.

The “ospf,” “tempospf” and “mainospf” tables are similar but contain different sources of the information. The “ospf” table contains the information of OSPF routes gathered from the router configuration files. The “tempospf” table contains additional routes added to an OSPF area that have been redistributed from either static or connected network prefixes. The “mainospf” table is the union of the “ospf” table and “tempospf” table which has all the OSPF routes for each OSPF area. The “ospf” table is unchanged after the parsing of router configuration files. The “tempospf” table is populated from the redistributed routes using the “redistribute” table. The reason for the separation of the parsing and derived information is to have the flexibility of not tainting the original information from router configuration files. The contents of the “ospf,” “tempospf” or “mainospf” tables consists of the router’s name, the OSPF label of the routing process, the IP address, the OSPF area, the OSPF link cost, the external administrative distance, the internal-area administrative distance, the intra-area administrative distance and the type of redistribution.

Table "public.ospf"

Column	Type
routername	text
ospflabel	text
ip	inet
ospfarea	integer
ospfcost	integer
externalad	integer
interareaad	integer
intraareaad	integer
typeredistribute	text

Table 5. Ospf specification.

Table "public.tempospf"

Column	Type
routername	text
ospflabel	text
ip	inet
ospfarea	integer
ospfcost	integer
externalad	integer
interareaad	integer
intraareaad	integer
typeredistribute	text

Table 6. Tempospf specification.

View "public.mainospf"

Column	Type
routername	text
ospflabel	text
ip	inet
ospfarea	integer
ospfcost	integer
externalad	integer
interareaad	integer
intraareaad	integer
typeredistribute	text

Table 7. Mainospf specification.

The “algorithm” table mirrors the algorithm described in [2]. The “algorithm” table represents the edges of the route propagation graph described in the background chapter. The “algorithm” table consists of the router’s name, the source router, the destination router, the administrative distance of the edge and whether the edge is active.

Table "public.algorithm"

Column	Type
routername	text
routefrom	text
routeto	text
ad	integer
active	text

Table 8. Algorithm specification.

The “ritable” table is primarily used to distinguish the interface’s OSPF area for each router. The “ritable” table consists of the router’s name, the OSPF area and the interface name that pertains to the OSPF area.

Table "public.ritable"

Column	Type
routername	text
instance	text
interface	text

Table 9. Ritable specification.

The “instancetab” table produces the router’s name and administrative distance by OSPF area. The “instancetab” table consists of the OSPF area, router’s name and the default administrative distance.

View "public.instancetab"

Column	Type
instanceid	integer
routername	text
defaultad	integer

Table 10. Instancetab specification.

The “morered” table is used for storing costs of specified redistribution routes from OSPF routers. The “morered” table plays an important role when calculating cost for redistributed connected subnets and static routes that traverse OSPF areas. The “morered” table consists of the router name, the source OSPF area, the destination OSPF area and the cost of the specified redistribution.

Table "public.morered"

Column	Type
routername	text
routefrom	text
routeto	text
cost	integer

Table 11. Morered specification.

The “output” table is used for the final construction of the router’s FIB table. The “output” table is queried to obtain each router’s FIB table. Most of the functions store their results into the “output” table. This table is used solely for the output of the finalized FIB table information. The “output” table is composed from the router’s name, the type of route (e.g. connected, static or OSPF), whether the OSPF redistribution is either internal or external, the IP address of the network prefix, the IP address of the interface, the name of the interface and the area of the OSPF routes.

Table "public.output"

Column	Type
routername	text
typeroute	text
ospfredistribute	text
ip	inet
interfaceip	inet
interface	text
area	text

Table 12. Output specification.

B. CODE

1. Parsing

There are four different database tables that hold the information parsed from the router configuration files: “interface,” “redistribute,” “ospf” and “static”. Each one of the routers has a unique router name for each router configuration processed because there has to be a unique identifier for each router configuration. Since there are four main parts of the router configuration, there are four different tables that contain individual components of the router configuration files, as shown in Figure 17.

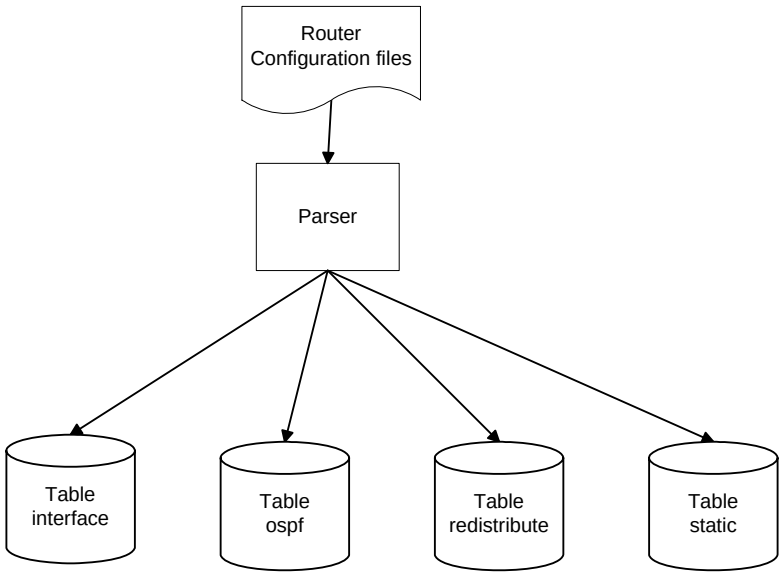

Figure 17. Tables populated from the parsing of router configuration files.

2. Static and Connected

Once the parsing of the router configuration files is completed, the next step is to discover the locally connected subnets and static routes. Figure 18 shows the flow of discovering locally connected subnets for each router. One of the unexpected points of discovering the locally connected subnets is that both endpoints on a link between routers will remain within the same network prefix. For example, if Router 1 has an IP address of 10.1.13.13/24 and connects to Router 3 which has an IP address of 10.1.13.31/24. Both Router 1 and Router 3 IP addresses are contained within the network prefix of 10.1.13.0/24. Thus, the "getConnected" function receives the locally connected subnets by querying the "interface" table for the unique network prefixes. Then, the function groups the connected subnets by network prefix and provide this information to the "getLoopback" function. Loopback is a unique connected subnet because the two endpoints of a loopback are contained within the same router. The "getLoopback" function looks for all of the specified loopback connections. Then, the "getLoopback" function inserts the loopback connections and results of the "getConnected" function into the "output" table as locally connected subnets.

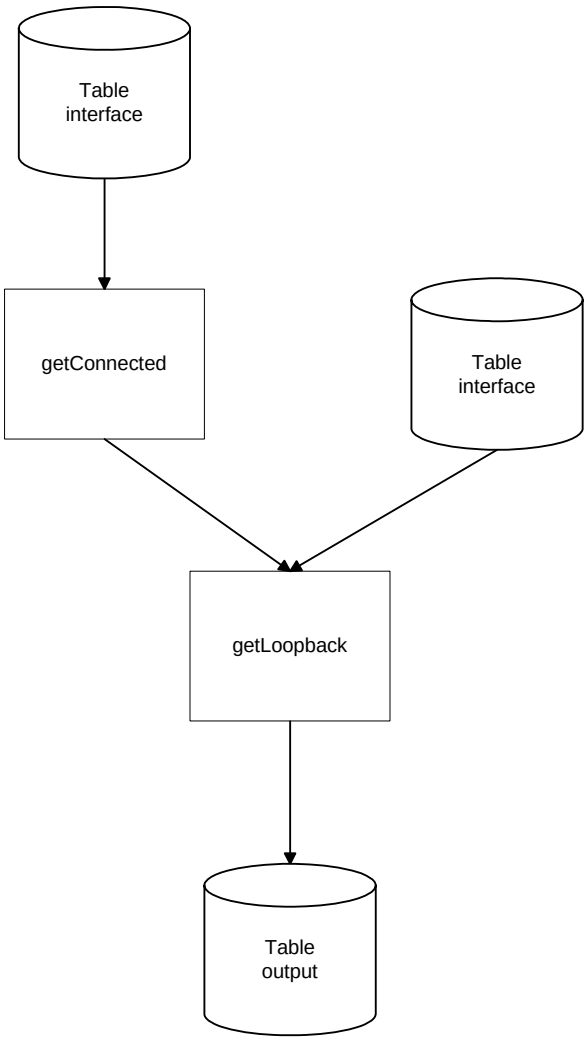

Figure 18. Tables used and populated for the “getConnected” and “getLoopback” function.

Static routes are less complicated than the locally connected subnets. Figure 19 shows a diagram of actions used to input static routes into the finalized FIB table for all routers. The “getStatic” function retrieves the static routes from the “static” table. Then, the “getStatic” function uses the information from the “static” table and inserts the static routes into the “output” table.

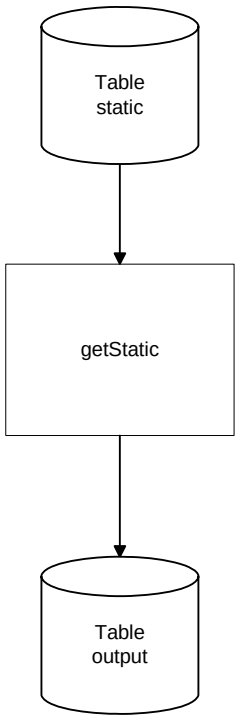

Figure 19. Tables used and populated for the “getStatic” function.

3. Redistributed Connected and Static

After the recognition of connected subnets and static routes, redistributed connected subnets and static routes are included into the appropriate OSPF routing process. In Figure 20, the insertions of redistributed connected subnets are performed by the “getRCOspf” function. The “getRCOspf” function utilizes information from the “redistribute,” “ospf” and “interface” tables. The “redistribute” table describes routers that are redistributing connected subnets into an OSPF area. The information gathered from the “redistribute” table is the router’s name, the OSPF label and the cost of the redistributed link. Then, the OSPF area and IP addresses of OSPF routes are collected from the “ospf” table according to the previously connected router name and OSPF label. The IP addresses used for connected subnets are collected from the “interface” table. All of the connected subnets are redistributed into the OSPF area if they are not already routes inserted into the “output” table.

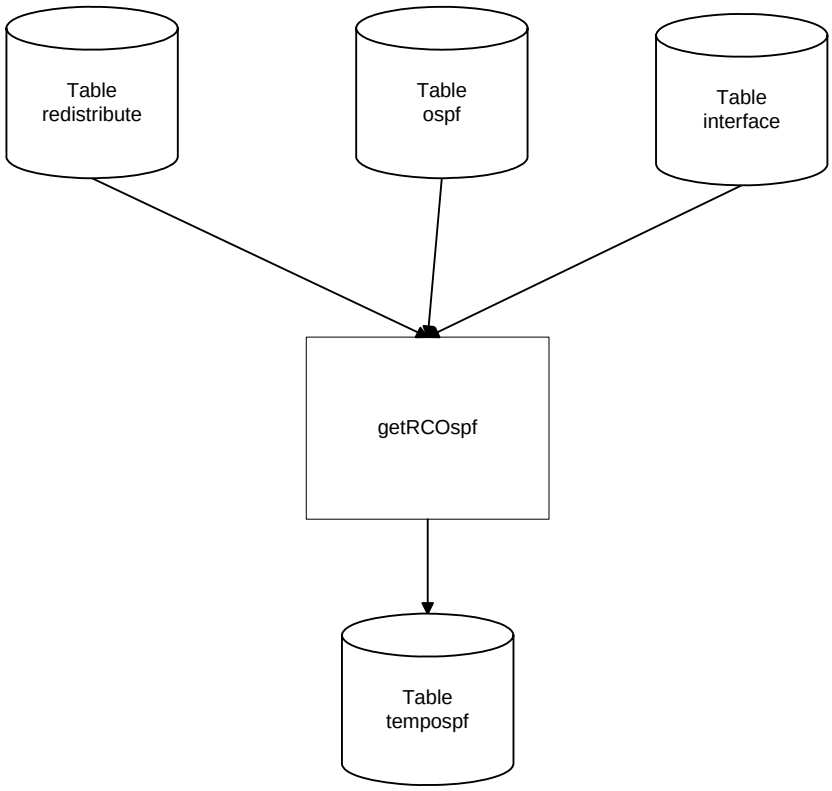

Figure 20. Tables used and populated for the “getRCOspf” function.

Figure 21 shows the flow diagram for the process of inserting redistributed static routes into an OSPF area from “redistribute,” “ospf” and “static” tables, which is encapsulated in the “getRSOspf” function. Since connected subnets are different than static routes, the static routes are stored in the “static” table instead of the “interface” table. Also, the “redistribute” table describes routers that are redistributing static routes into an OSPF area. The information gathered from the “redistribute” table is the router’s name, the OSPF label and the cost of the redistributed link. Then, the OSPF area membership is collected from the “ospf” table along with the router’s name and OSPF label. The IP addresses for static routes are collected from the “static” table and redistributed into the OSPF area.

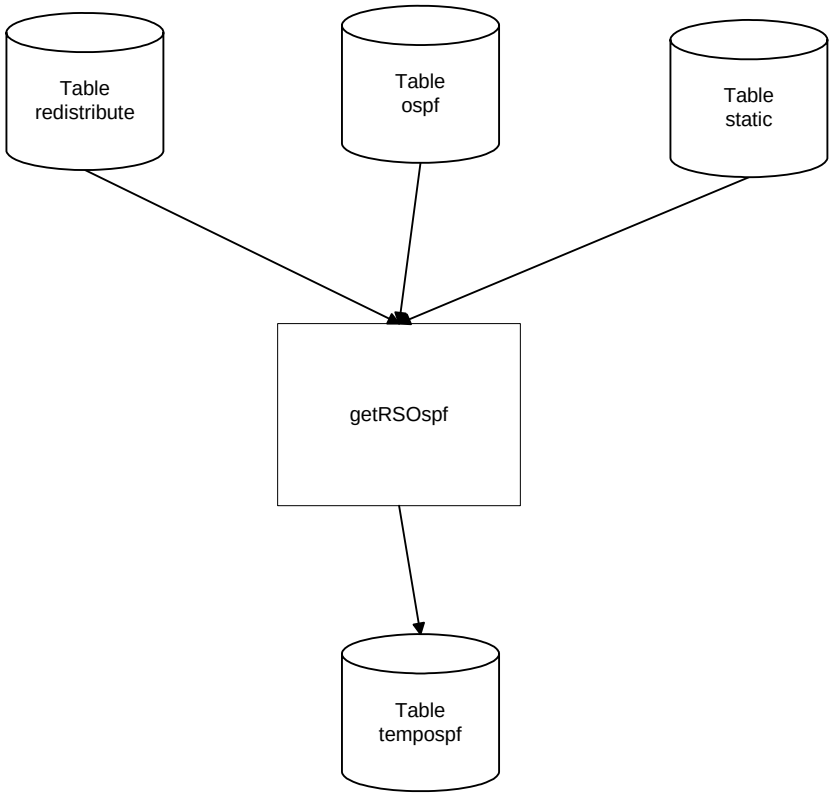

Figure 21. Tables used and populated for the “getRSOspf” function.

4. OSPF Single Area

Single area OSPF analysis is the first case of combining large groups of network prefixes. When network prefixes are grouped into one OSPF area, network administrators can manage their network topology without manual configuration, unlike connected and static configurations. Single area OSPF analysis has been broken into three different functions: “getOspf,” “pruner” and “exex”. Figure 22 shows which tables play a role for inputs and outputs to the “getOspf” function. The “getOspf” function collects all possible OSPF routes that are relevant for each OSPF area. The first step is to retrieve all of the OSPF areas within the network from the “mainospf” table. Once the OSPF areas within the network are known, the system does the following steps for each OSPF area. The system collects IP addresses and routers’ names for all the routers within an OSPF area from the “mainospf” and “ospf” tables. Dijkstra’s algorithm computes the shortest paths to the previously collected network prefixes. Since Dijkstra’s algorithm

requires the network to be represented as a graph, the system creates a table that represents a graph as edges and nodes using data from the “interface” and “ospf” tables. The edges represent the OSPF cost while the nodes are identified with the router’s name. The “tempospf” table adds edges going from a known node to a newly created node to display static or connected redistribution. Dijkstra’s algorithm connects all nodes in the graph using the lowest cost connections, as well as it guarantees no loops in the graph. This forces a particular implementation of static and connected redistribution mechanism, as loopback routes would be recognized as loops. For example, a static route has been redistributed into an OSPF area with a router name of Router5. The two nodes inserted into the table will be Router5 and Router5s<counter>. The counter will be incremented every time a new redistributed static or connected node is inserted into the Dijkstra graph. After the Dijkstra graph has been created, each router will run Dijkstra’s algorithm for all IP addresses that are not directly connected to the router currently running Dijkstra’s algorithm. When a new OSPF route is discovered, the program inserts the route into the “output” table.

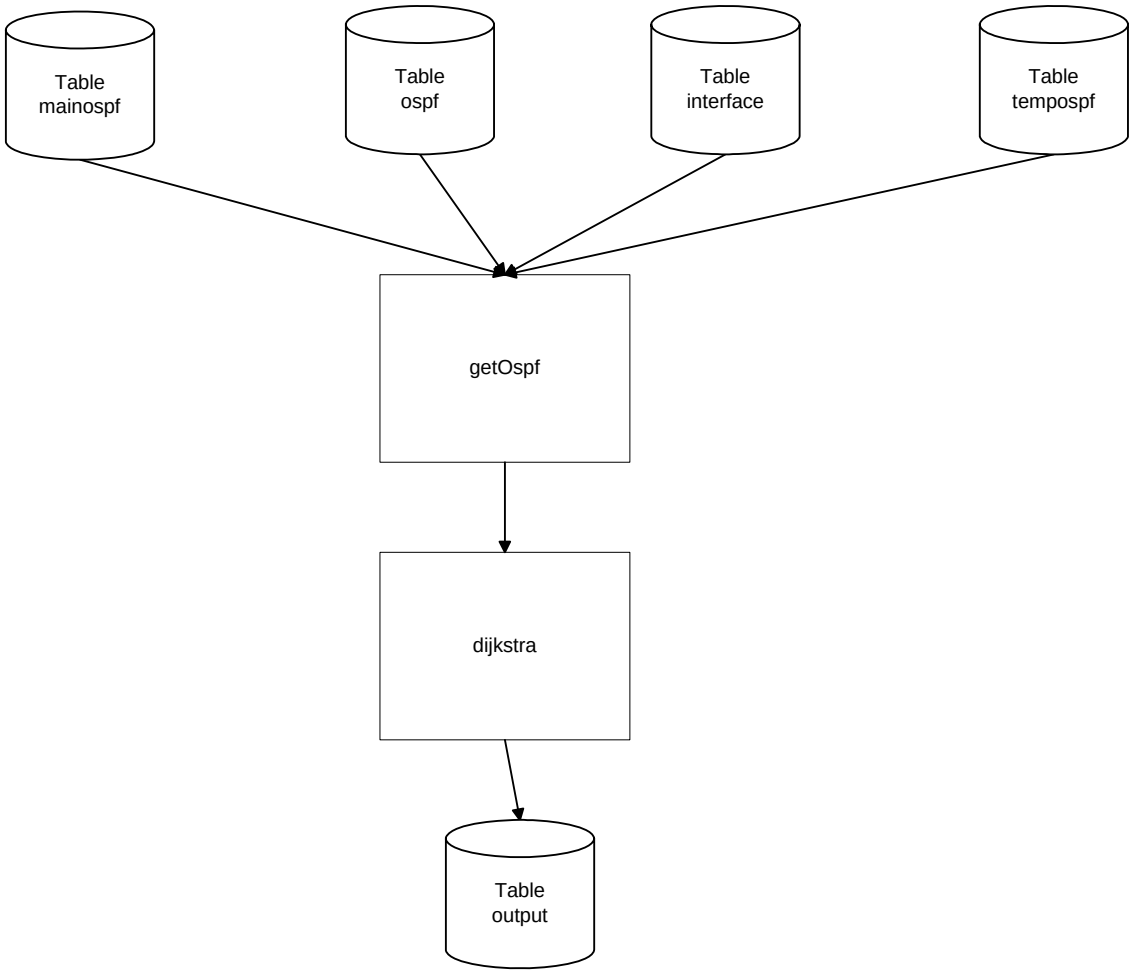

Figure 22. Tables used and populated for the “getOspf” function.

The “pruner” function, as shown in Figure 23, removes all OSPF routes from the “output” table that have been previously labeled as connected and static routes. Connected and static routes have a higher precedence over OSPF routes. Thus, the “pruner” function goes through every router and removes any OSPF route that has a corresponding static or connected route.

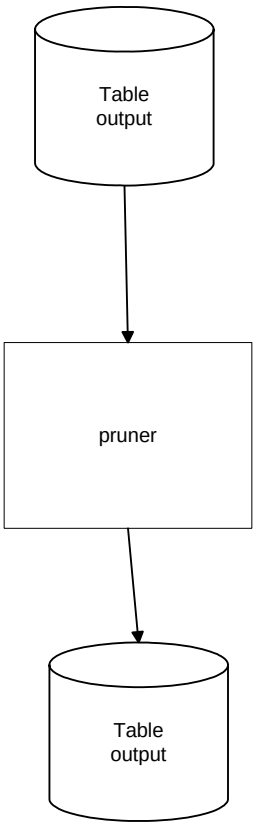

Figure 23. Tables used and populated for the “pruner” function.

The “exex” function, shown in Figure 24, labels redistributed routes from static or connected to become external OSPF routes in the “output” table. The system goes through every router and changes any OSPF route redistributed into a router from static or connected to an external route.

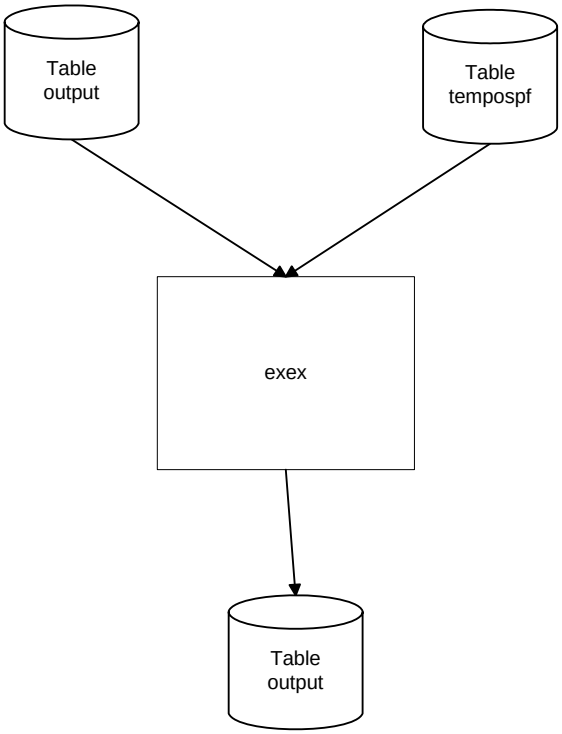

Figure 24. Tables used and populated for the “exex” function.

5. OSPF Multiple Area

The last part of OSPF analysis implements mutual redistribution between OSPF areas. Mutual redistribution requires full understanding of all the previously discussed functions. Mutual redistribution has been broken into five different functions: “getROOspf,” “setRed,” “getWeight,” “getIntRedExt” and “getIntRedInt”. Figure 25 demonstrates how the system implements the first part of mutual redistribution between OSPF areas, which is the “getROOspf” function. The system prepares the routing information for the algorithm [2] from the “redistribute,” “mainospf,” “ospf” and “interface” tables. The graph section of the algorithm is inserted into the “algorithm” table from the information gleaned from the “redistribute” and “ospf” tables. The next part of the system contains a record of interface names and interface IP addresses for the routers on the network, which is stored in the “ritable” table. Once the previous information is gathered, the algorithm [2] runs to find a way to redistribute information between OSPF areas.

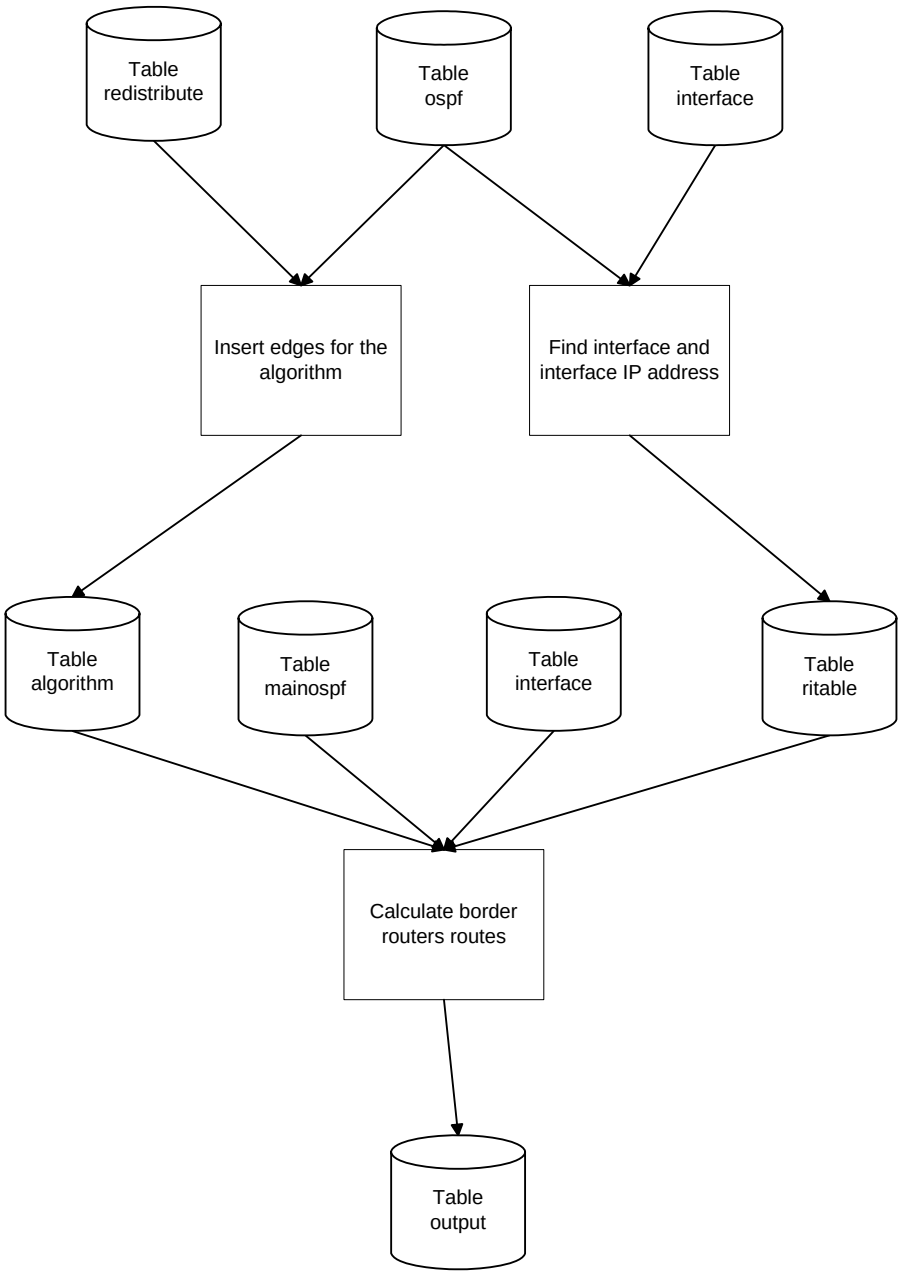

Figure 25. Tables used and populated for the the “getROOspf” function.

Once the algorithm finishes, the next part of mutual redistribution will be finding the specified costs for redistributed OSPF routers. The specified costs have a higher precedence than calculating the cost by tracing through the path. Figure 26 shows a flow

diagram for finding the lowest redistributed cost from routers, which is the "setRed" function. If a router sends a LSA packet with a lower cost to a particular OSPF area, the other routers in that OSPF area will agree to use that cost. For example, Router 1 in Figure 8 announces a cost of 100 to any OSPF area 1 network prefixes. Router 2 will remember the cost of 100 to any OSPF area 1 network prefix through Router 1.

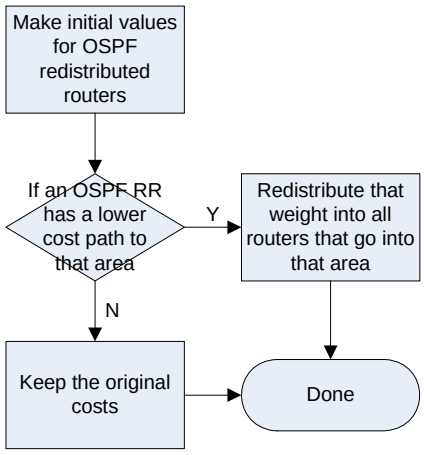

Figure 26. The flow diagram for the "setRed" function.

Figure 27 and 28 shows a flow diagram of how to calculate the cost for every path in the "output" table, which is the "getWeight" function. This function has three parameters: the starting router name, the destination network IP address and the starting interface IP address. This function uses the information from the setRed flow diagram to remember the lowest redistributed OSPF cost for an OSPF area. Figure 27 displays the flow diagram for obtaining the cost of an internal network prefix to an OSPF area. The "getWeight" function gathers the three parameters and sets the total cost to zero. While the router does not have the destination network prefix as a directly connected subnet or static route, the OSPF link cost to the destination prefix will be added to the total cost. The next router analyzed will be determined by the FIB table of the previous router to reach the destination network prefix because Dijkstra's algorithm has been used to calculate the path for each route to every network prefix within an OSPF area in the single area OSPF section of the paper. Once the destination network prefix is attached to the current router, the OSPF link cost of the destination network prefix will be added to

the total cost. Figure 28 displays the flow diagram for obtaining the cost of an external network prefix to an OSPF area. The “getWeight” function gathers the three parameters and sets the cost to zero. If the destination network prefix is a redistributed connected subnet or static route and the cost is specified, than the lowest redistributed specified cost is used as the total cost. If any of the OSPF redistributed routers have a specified cost in the direction of the path taken to reach the destination network prefix, the specified cost for the OSPF redistributed routers is used as the total cost. Otherwise, the cost will be the obtained from calculating the internal cost from the last border router.

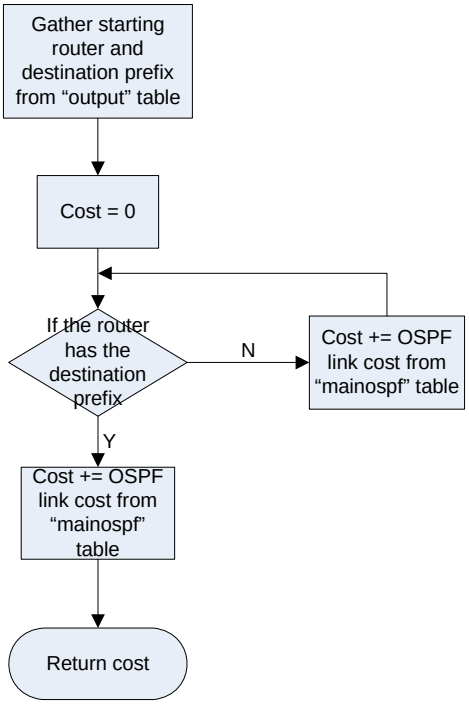

Figure 27. The flow diagram for the “getWeight” function for internal routes.

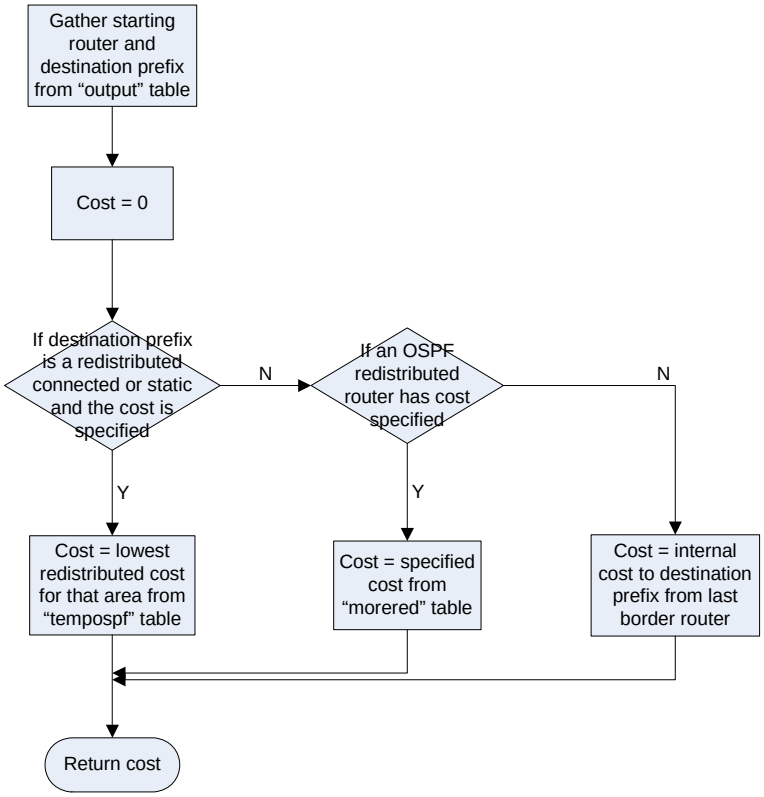

Figure 28. The flow diagram for the "getWeight" function for external routes.

Figure 29 and Figure 30 are similar but Figure 29 analyzes internal routes discovering external routes, which is the "getIntRedExt" function, where Figure 30 analyzes internal routes discovering better internal routes, which is the "getIntRedInt" function. The routes are found by looking at routers directly connected to OSPF redistributed routers and using the "getWeight" function to insert the lowest cost routes into the "output" table. Once the routes have been found for those routers, the routers connected to the previously analyzed routers become the new analyzed routers. The reason for this procedure is that the "getWeight" function finds a new cost from tracing the "output" table. If a route has not been specified by a router, the cost would resemble the cost of a loop since the function has ended without finding an adequate route. All of the routes found in Figure 29 will be labeled as external routes because the routes are going to border routers that will cross an OSPF area. The routes found in Figure 30 will be labeled as external routes if the lowest cost route is not the original route found using the single area OSPF calculation.

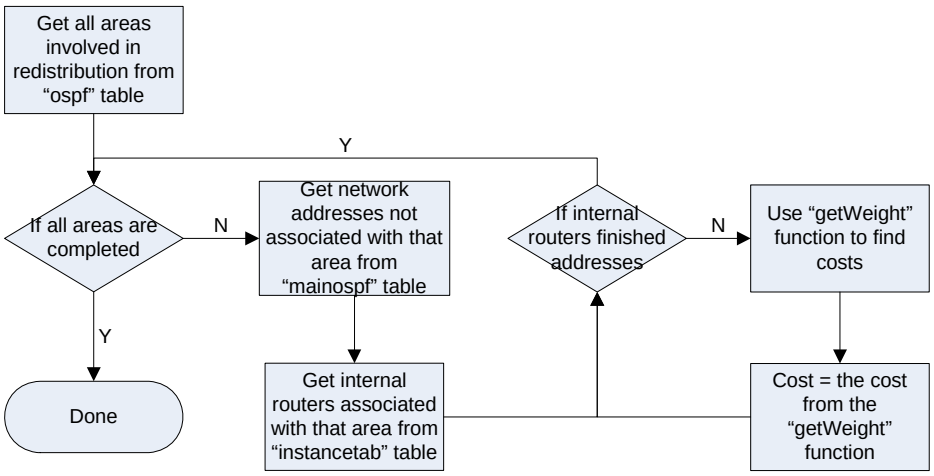

Figure 29. The flow diagram for the "getIntRedExt" function

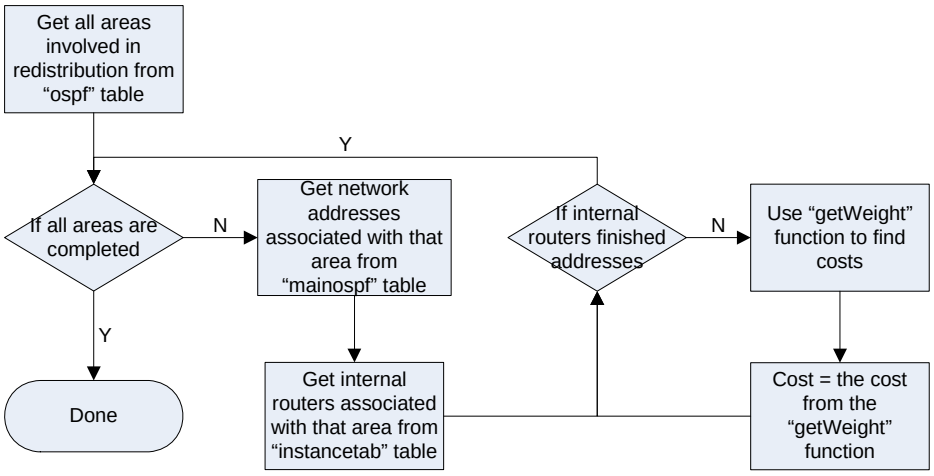

Figure 30. The flow diagram for the "getIntRedInt" function.

6. Printing Output

The "printOutput" function prints the output from each router's FIB into a Cisco FIB table format that is produced from Cisco routers. Figure 31 shows a flow diagram for outputting the router's FIB table to Cisco's format. The types of routes printed are connected, static, internal OSPF and external OSPF routes. This function uses the "ospf" table to find the correct administrative distance and the "getWeight" function to calculate the cost for the route.

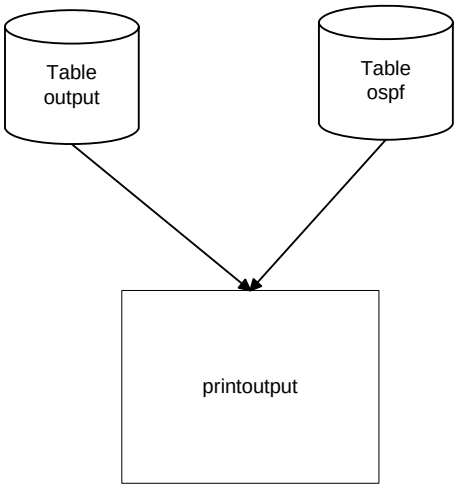

Figure 31. Tables used and populated for the the “printOutput” function.

THIS PAGE INTENTIONALLY LEFT BLANK

V. VALIDATION

The validation of the following experiments in this chapter proves to be beneficial for an in depth understanding of OSPF. Without a good base experiment layout, the validation of the program would be futile. The following three experiments were used to validate the program. All of the validations have been run through the program and compared by hand to the original "show ip route" files that are computed by the Cisco routers. The program accurately resembles the "show ip route" files from the Cisco routers. All of the experiments were run in the same manner except changing the router configuration input file in TestDriver.java. For all the following steps, it is assumed that the user will be in the main directory of the project. The first step is to clear the database of any previous information, achieved by the "psql -f cleaner" command. The next step is to compile all the Java files, with the "javac *.java" command. The parsing file is called TestDriver and to run the compiled TestDriver file, use the command "java -classpath /home/mcmanst/code/postgresql-8.1-404.jdbc2.jar:/home/mcmanst/code TestDriver". The classpath is changed because the JDBC driver has to also be specified to connect to the database. The static analysis file, FindRoute, is run by using the "java -classpath /home/mcmanst/code/postgresql-8.1-404.jdbc2.jar:/home/mcmanst/code FindRoute" command.

A. EXPERIMENT 1

The first experiment has two different OSPF areas that do not redistribute any OSPF routes between them. This experiment verifies that connected, static, and single area OSPF routes are accurately simulated within the network. The router configuration files for experiment 1 and "show ip route" files are in Appendix F and I, respectively. In Figure 32, OSPF area 1 is analyzed. The link costs are annotated on each edge, symmetric by default and indicated by an arrow when customized. The administrative distances of the border OSPF processes are the values underlined in the graph. The unique properties of the setup from Figure 32 are redistributed connected subnets and static routes that have informed OSPF area 1 of their existence. This scenario gives a

deeper understanding of how redistributed static routes and connected subnets compute the cost of the path to those network prefixes. Since all the costs for redistributed static routes and connected subnets have been specified in the router configuration files, the cost to the network prefix is the cost of the explicitly specified redistributed route. An unusual side effect of this philosophy is that OSPF will prefer redistributed routes over connected if the cost has been specified in the router's configuration file.

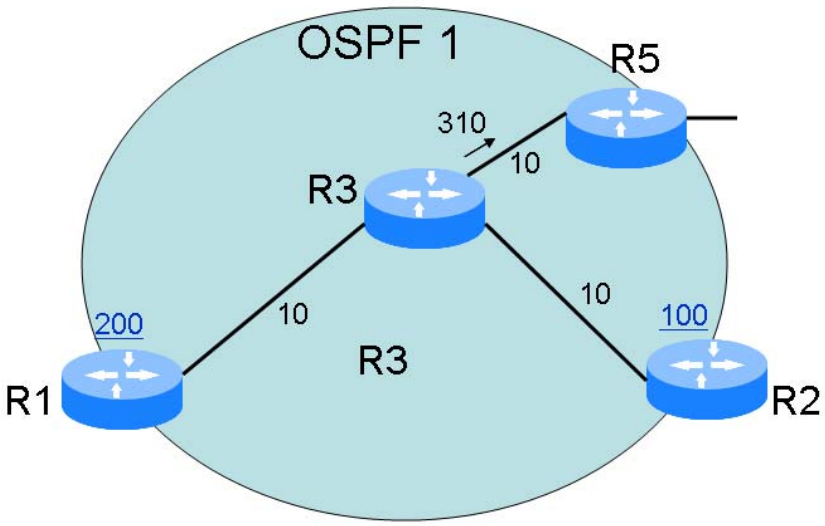

Figure 32. Experiment 1 with OSPF area 1.

Figure 33 shows a usual understanding of how OSPF routers would find a cost for a network prefix. There is no redistribution of any routes within Figure 33. There are no OSPF routes from R4 because the network prefixes are directly connected to R4. If a network prefix is known to a router as directly connected or static route, the OSPF characteristics are ignored because static and connected routes have a higher preference than OSPF routes. The output from experiment 1 is in Figure 34 which is the same as Appendix I.

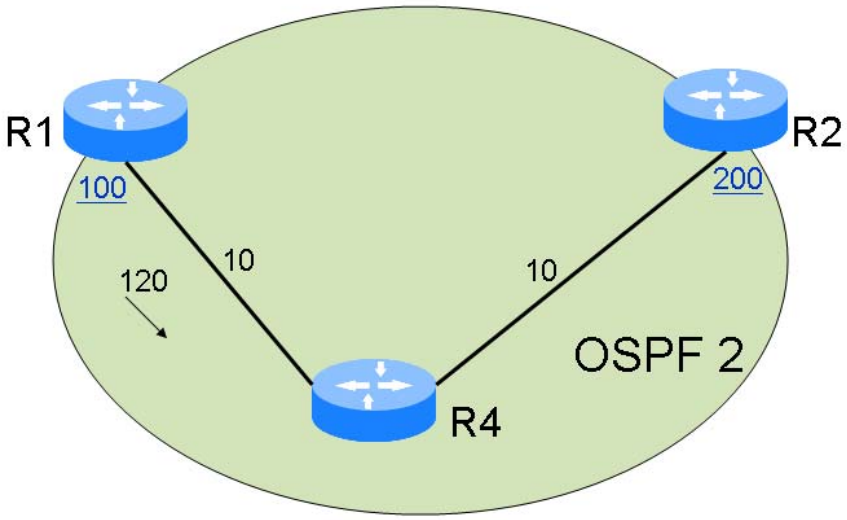

Figure 33. Experiment with OSPF area 2.

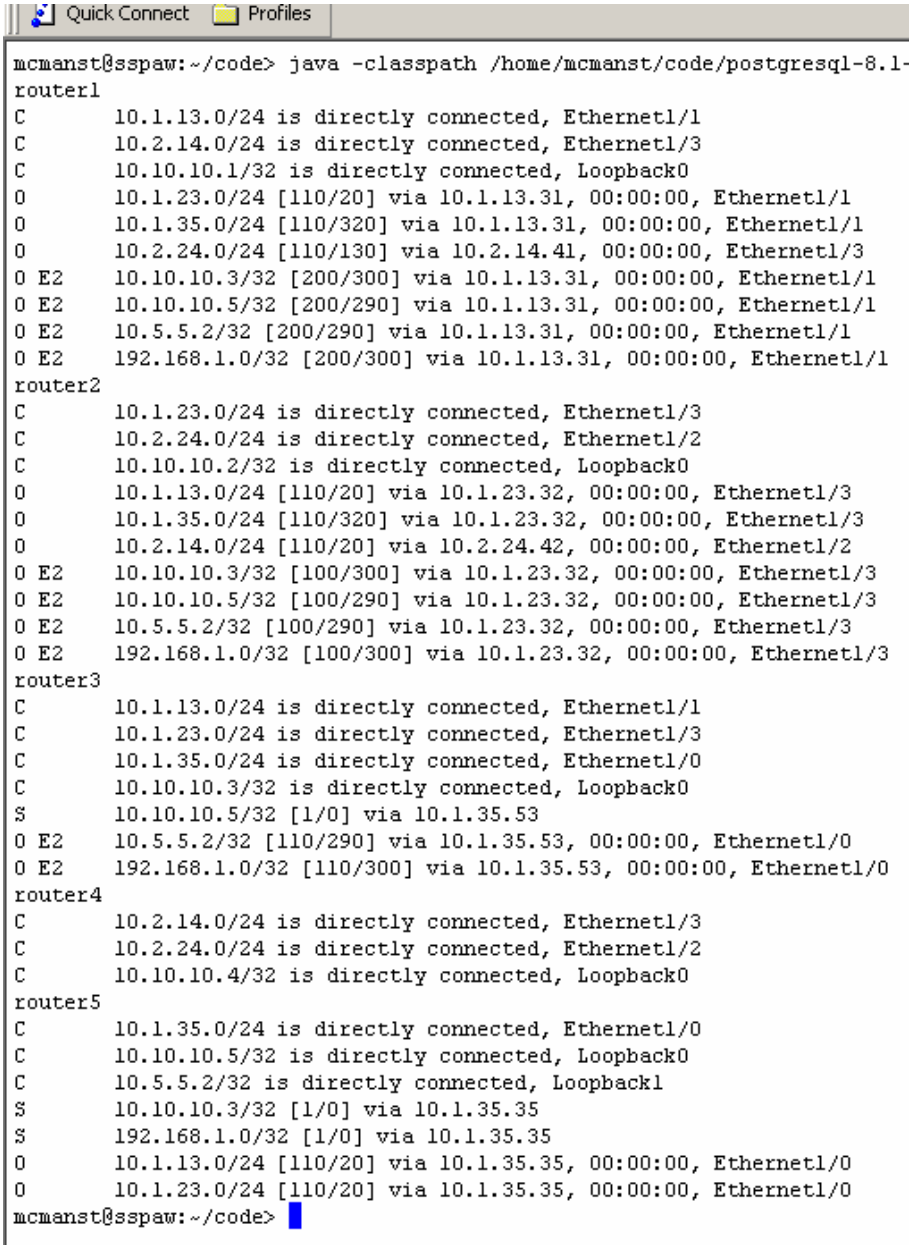

Figure 34. Output from program for experiment 1.

B. EXPERIMENT 2

Experiment 2, as shown is Figure 35, was the original network configuration used to validate the program. The router configuration files for experiment 2 and "show ip route" files are in Appendix G and J, respectively. This experiment includes mutual redistribution between OSPF area 1 and 2 activated on routers R1 and R2. The costs of

redistribution on R1 and R2 are shown with an arrow pointing in the direction of the redistribution. This experiment gives a better understanding of redistribution and calculating costs from routers to network prefixes. When a redistributed cost has been specified for a network prefix, it has a higher priority than any other cost on this router for the target prefix. For example, R3 will use the cost of 100 for redistributed routes that are static and connected because R1 and R2 informed R3 that the costs for those redistributed routes are 100 through LSA packets. When handling routes are not redistributed between two different areas, the cost calculations for paths are set to zero. This was especially noticeable going from R1 to the network prefix between R2 and R3. The cost from the "show ip route" files is 10. The reason for this is because the cost calculation gets reset at R2 and the calculated cost of the link becomes 10. The output from experiment 2 is in Figure 36 which is the same as Appendix J.

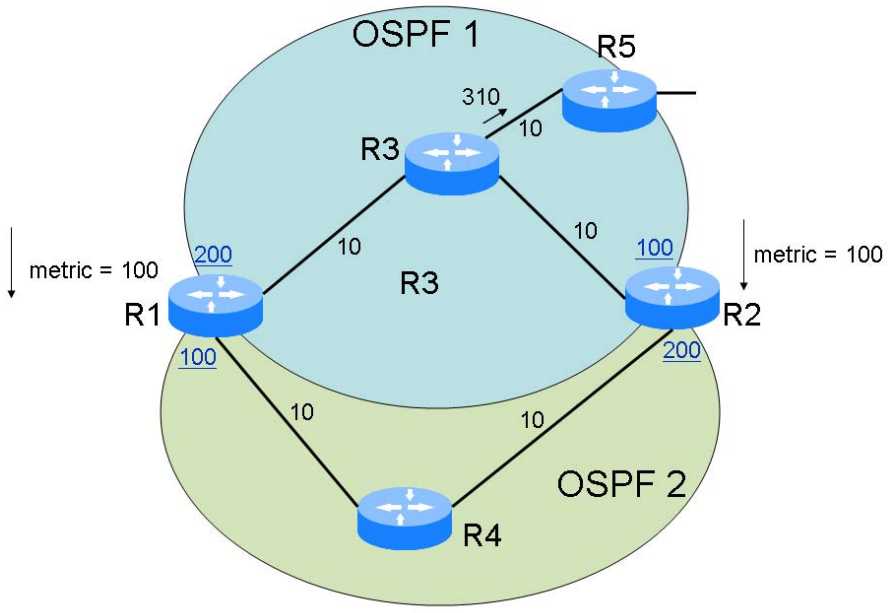

Figure 35. Experiment 2.

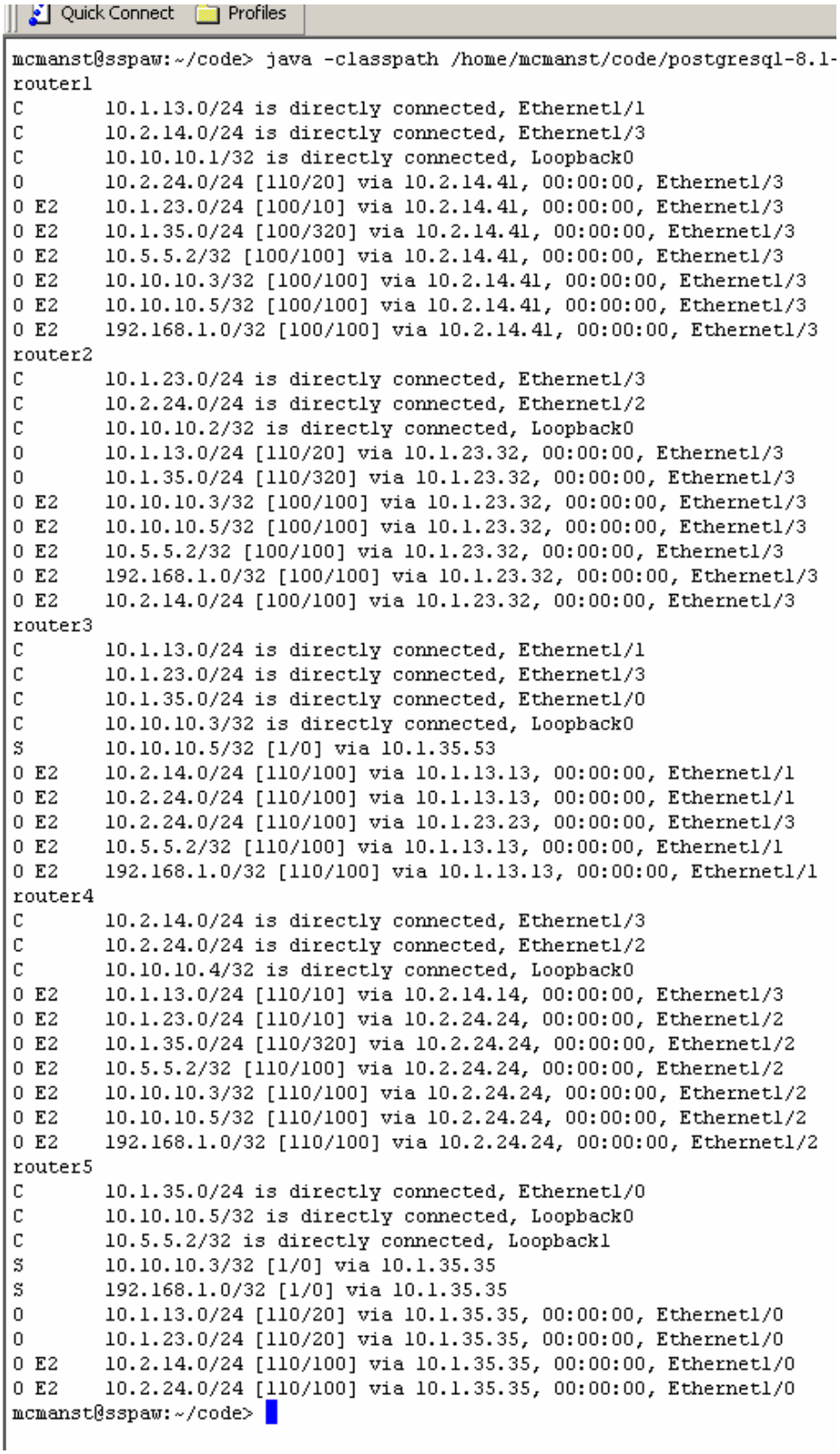

Figure 36. Output from program from experiment 2.

C. EXPERIMENT 3

Experiment 3 is a further confirmation of the issues brought up in experiment 2, which is shown in Figure 37. The router configuration files for experiment 3 and "show ip route" files are in Appendix H and K, respectively. The only difference between experiment 2 and 3 is that the cost for both directions of R2 are specified for redistribution in experiment 3. For any destination network prefix from R4, the cost will be 10 because the metric was specified for R2 going from OSPF area 2 to OSPF area 1. The output from experiment 3 is in Figure 38 which is the same as Appendix K.

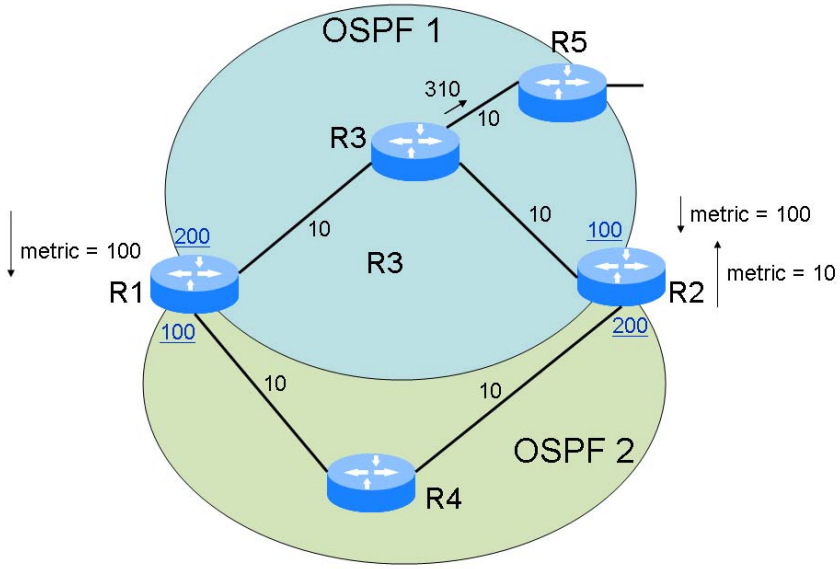

Figure 37. Experiment 3.

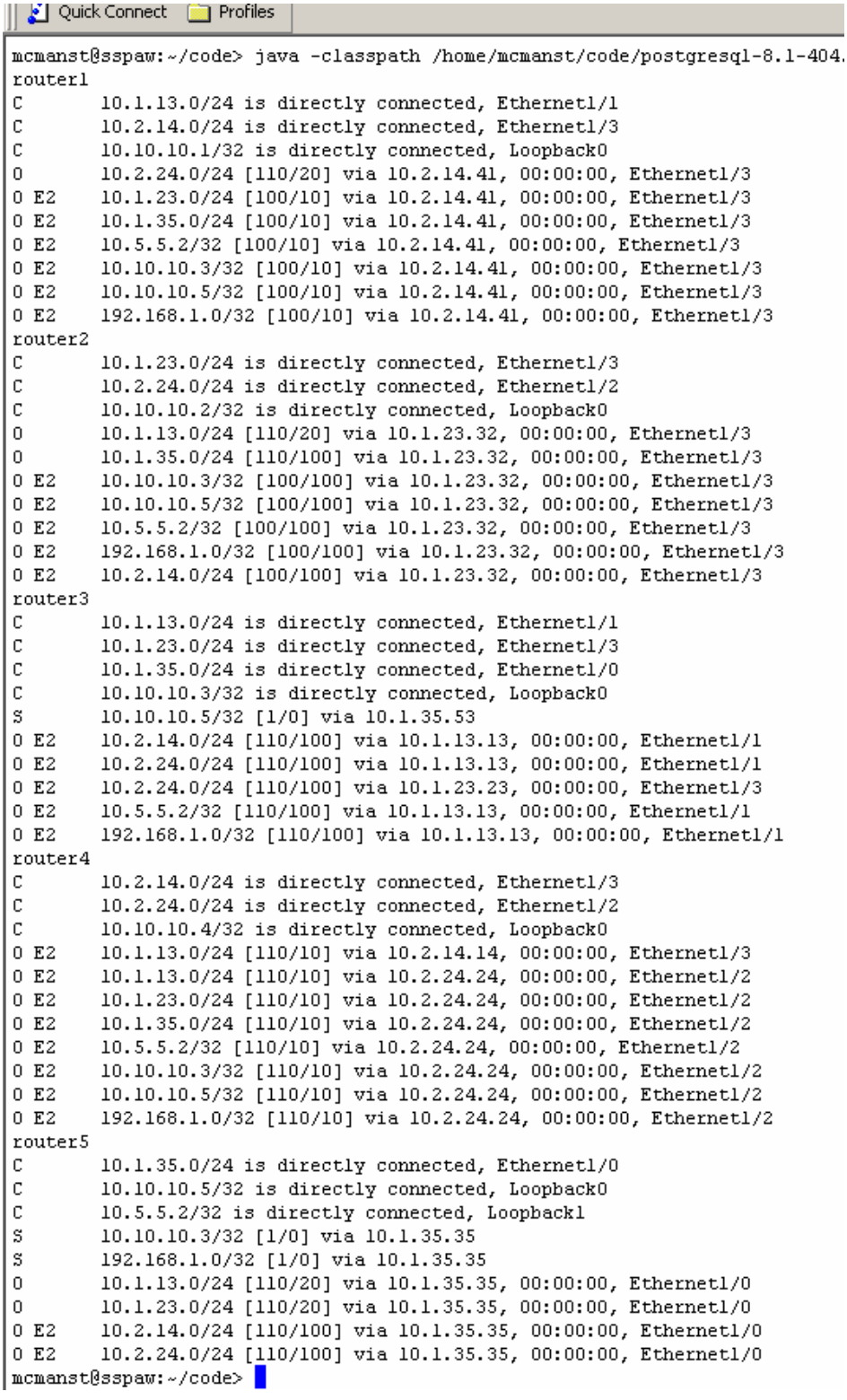

Figure 38. Output from program on experiment 3.

VI. CONCLUSIONS

Overall the thesis is successful in addressing some of the research questions listed in Chapter I. The unusual nature of redistributed static routes and connected subnets into OSPF demonstrated in experiment 2 and 3 bring to light many further questions of other routing protocols interacting with OSPF, e.g. Border Gateway Protocol (BGP). Also, the experience gained in architecting and implementing this system has resulted in some unexpected insights into the OSPF routing protocol. This chapter first summarizes these insights and then discusses possible improvements and future work for this system.

A. FINDINGS

The major findings from this research effort are:

- Static analysis is feasible by using router configuration files as the source code of a distributed program whose execution determines the host level reachability of the network.
- Computing the cost for OSPF routes from internal routers of an OSPF area was harder than expected.
- Using a database to store the data has greatly simplified the programming of this system.

B. RECOMMENDATIONS FOR FUTURE WORK

The next logical step is to include the static analysis of BGP. Many new challenges may arise when dealing with the interactions between BGP and OSPF. However, the addition of BGP will allow the system to perform static analysis of a large class of networks where routing is carried out by both BGP backbone routers and OSPF edge routers.

Another feature that should be added to the system is the ability to infer the effect of the OSPF load balancing feature. Load balancing in OSPF is implicit, not done by physically specifying the links to use for splitting the load. To set load balancing for OSPF, a network administrator will have to ensure that (i) the network would produce two or more OSPF routes with the same administrative distance but different outbound

interfaces, and (ii) the cost to reach the target network prefix would have to be the same for all the routes. Therefore, the configuration of OSPF load balancing is an error prone task and as such should benefit greatly from the static analysis.

APPENDIX A. TESTDRIVER.JAVA

```
//Stephen McManus
//Parser of the configuration files
import java.sql.*;
import java.util.*;
import java.io.*;

public class TestDriver{
	private Connection db; //database connection
	private BufferedReader in; //input file
	private String RN; //router name
	private String HN; //hostname
	private String sqltext; //sqltext
	private int rnum;
	private List<String> ipaddred; //interface ip for ospf cost
	private List<String> weighter; //weight for ospf cost
	private Statement state; //inject sql queries
	private Statement state2; //inject sql queries

	public static void main(String args[]) throws Exception{
		Class.forName("org.postgresql.Driver");
		TestDriver test = new TestDriver();

	    // Enter host ip, database name, username, password
	    System.out.println("Input the database info first.");

		String ip = test.getInput("  IP address of host machine: ");
		String name = test.getInput("  Database name: ");
	    String owner = test.getInput("  Owner: ");
		String password = test.getInput("  Password: ");

	    // an example would be:
	    // "172.20.106.36", "mcmanst","mcmanst","fluffy";
		test.Connector(ip, name, owner, password); //Connect to the database
		test.Parse(); //Parse the router configuration files
	}

	public void Parse() throws Exception{
		ipaddred = new ArrayList<String>();
		weighter = new ArrayList<String>();
		state = db.createStatement(); //Be able to make sql queries later
		state2 = db.createStatement(); //Be able to make sql queries later

	    // Enter name for the ASCII file containing all configuration files
		String configsFile = getInput("What is the network to parse: ");
	    // an example: configs2 -- Experiment 2
		in = new BufferedReader(new FileReader(configsFile)); //New configuration
files
		String str;
		rnum = 1;
		RN = "router"+rnum; //Name of the router
		while((str = in.readLine()) != null){
			str = str.trim();
			if((str.startsWith("interface"))){
				str = interfaceParse(str); //Interface parse
			}
			else if((str.startsWith("hostname"))){
				str = hostParse(str); //Hostname parse
			}
			else if((str.startsWith("router ospf"))){
				str = ospfParse(str); //OSPF parse
			}
			else if((str.startsWith("ip route"))){
				str = staticParse(str); //Static parse
			}
```

```
                        else if((str.startsWith("end"))){
                                if(ipaddred.isEmpty()!=true){ //if interface had the ospf
cost instead of in router ospf
                                        for(int i=0;i<ipaddred.size();i++){
                                                sqltext = "update ospf set ospfcost =
'"+weighter.get(i)+"' where ip <<= '"+ipaddred.get(i)+"' and routername = '"+RN+"'";
                                                state.executeUpdate(sqltext);
                                        }
                                        ipaddred.clear();
                                        weighter.clear();
                                }
                                rnum++; //Change router name
                                RN = "router"+rnum;
                        }
                }
        }

        public String staticParse(String str) throws Exception{
                List<String> olist = new ArrayList<String>();
                String nstr,inip,outip,outip2;
                ResultSet rs,rs2;
                outip = "null";
                int indexed,nm;
                nstr = skipper(str);
                nstr = skipper(nstr);
                indexed = nstr.indexOf(" ");
                inip = nstr.substring(0,indexed); //IP address trying to reach
                nstr = skipper(nstr);
                indexed = nstr.indexOf(" ");
                nm = netmasker(nstr.substring(0,indexed));
                inip+="/"+nm;
                nstr = skipper(nstr);
                if((nstr.matches(".*[\\.]{1}.*[\\.]{1}.*[\\.]{1}.*")) == false){ //If not
an ip address
                        sqltext = "select host(ip),network(ip) from interface where
interfacename = '"+nstr+"' and routername = '"+RN+"'";
                        rs = state.executeQuery(sqltext);
                        if(rs.next()){
                                outip = rs.getString(1); //Which ip address to go out
                                outip2 = rs.getString(2);
                                sqltext = "select host(ip) from interface where host(ip) !=
'"+outip+"' and network(ip) <<= '"+outip2+"'";
                                rs2 = state2.executeQuery(sqltext);
                                if(rs2.next()){
                                        outip = rs2.getString(1);
                                        sqltext = "insert into static
values('"+RN+"','"+inip+"','"+outip+"')";
                                        olist.add(sqltext);
                                }
                        }

                }
                else{ //If an ip address
                        outip = nstr; //Which ip address to go out
                        sqltext = "insert into static
values('"+RN+"','"+inip+"','"+outip+"')";
                        olist.add(sqltext);
                }
                for(int i=0;i<olist.size();i++)
                        state.executeUpdate(olist.get(i));
                return nstr;
        }

        public String interfaceParse(String str) throws Exception{
                String nstr,intn,ip,weighted;
                int indexed;
                int nmbits;
                boolean minimized=false;
                nstr = skipper(str);
```

```
                intn = nstr; //Interface name
                nstr = in.readLine();
                nstr = nstr.trim();
                weighted = "10"; //OSPF cost
                if(intn.matches("Ethernet.*")==true)
                        weighted = "10";
                if(intn.matches("FastEthernet.*")==true)
                        weighted = "1";
                ip = ""; //IP address
                while(nstr.matches("!") == false){
                        if(nstr.startsWith("ip address")){
                                nstr = skipper(nstr);
                                nstr = skipper(nstr);
                                indexed = nstr.indexOf(" ");
                                ip = nstr.substring(0,indexed);
                                nstr = skipper(nstr);
                                nmbits = netmasker(nstr);
                                ip+="/"+nmbits;
                                sqltext = "insert into interface
values('"+RN+"','"+HN+"','"+intn+"','"+ip+"')";
                        }
                        //OSPF cost
                        if(nstr.startsWith("ip ospf cost")){
                                nstr = skipper(nstr);
                                nstr = skipper(nstr);
                                nstr = skipper(nstr);
                                weighted = nstr;
                        }
                        if(nstr.startsWith("shutdown") || nstr.startsWith("no ip address"))
//if shutdown do not add
                                minimized=true;
                        if((nstr = in.readLine()) == null)
                                break;
                        else
                                nstr = nstr.trim();
                }
                if(minimized==false){
                        ipaddred.add(ip);
                        weighter.add(weighted);
                        state.executeUpdate(sqltext);
                }
                return nstr;
        }

        // Get the number of netmask bits out of the netmask
        public int netmasker(String str) throws Exception{
                String nstr,bstr;
                int indexed,i,num,nmbits=0;
                nstr = str + ".";
                for(i=0;i<4;i++){
                        indexed = nstr.indexOf(".");
                        bstr = nstr.substring(0,indexed);
                        num = Integer.parseInt(bstr);
                        bstr = Integer.toBinaryString(num);
                        nmbits += bstr.lastIndexOf("1")+1;
                        nstr = nstr.substring(indexed+1);
                }
                return nmbits;
        }

        // Get the number of netmask bits out of the hostmask
        public int hostmasker(String str) throws Exception{
                String nstr,bstr;
                int indexed,i,num,nmbits=0;
                nstr = str + ".";
                for(i=0;i<4;i++){
                        indexed = nstr.indexOf(".");
                        bstr = nstr.substring(0,indexed);
                        num = Integer.parseInt(bstr);
```

```
                        bstr = Integer.toBinaryString(num);
                        nmbits += 8 - (bstr.lastIndexOf("1") + 1);
                        nstr = nstr.substring(indexed+1);
                }
                return nmbits;
        }

        public String hostParse(String str) throws Exception{
                String nstr;
                nstr = skipper(str);
                HN = nstr; //hostname
                nstr = in.readLine();
                nstr = nstr.trim();
                while(nstr.matches("!") == false){
                        if((nstr = in.readLine()) == null)
                                break;
                        else
                                nstr=nstr.trim();
                }
                return nstr;
        }

        public String ospfParse(String str) throws Exception{
                List<String> olist = new ArrayList<String>();
                String nstr,nnstr,temp,ospfnum,ip,area,weight,exter,intera,intraa,changer;
                exter = "110";
                intera = "110";
                intraa = "110";
                ip = "null";
                area = "-1";
                weight = "1";
                int indexed,nm;
                nstr = skipper(str);
                nstr = skipper(nstr);
                ospfnum = nstr;
                nstr = in.readLine();
                nstr = nstr.trim();
                while(nstr.matches("!") == false){
                        if(nstr.startsWith("redistribute")){ //If redistribution
                                nstr = skipper(nstr);
                                indexed = nstr.indexOf(" ");
                                changer = "n";
                                if((nstr.substring(0,indexed).matches("static"))){ //If
static
                                        String other_ospf,oweight;
                                        oweight = "1";
                                        other_ospf = "null";
                                        nstr=skipper(nstr);
                                        nstr=skipper(nstr);
                                        indexed = nstr.indexOf(" ");
                                        oweight = nstr.substring(0,indexed);
                                        changer = "y";
                                        sqltext = "insert into redistribute
values('"+RN+"','"+ospfnum+"','static','"+ospfnum+"','"+oweight+"','"+changer+"')";
                                        state.executeUpdate(sqltext);
                                }
                                else if((nstr.substring(0,indexed).matches("connected"))){
//If connected
                                        String other_ospf,oweight;
                                        oweight = "1";
                                        other_ospf = "null";
                                        nstr=skipper(nstr);
                                        nstr=skipper(nstr);
                                        changer = "y";
                                        indexed = nstr.indexOf(" ");
                                        oweight = nstr.substring(0,indexed);
                                        sqltext = "insert into redistribute
values('"+RN+"','"+ospfnum+"','connected','"+ospfnum+"','"+oweight+"','"+changer+"')";
                                        state.executeUpdate(sqltext);
```

```
                        }
                        else if((nstr.substring(0,indexed).matches("ospf"))){ //If
ospf
                                String other_ospf,oweight;
                                oweight = "10";
                                nstr = skipper(nstr);
                                indexed = nstr.indexOf(" ");
                                other_ospf = nstr.substring(0,indexed);
                                nstr = skipper(nstr);
                                if(nstr.startsWith("metric")){
                                        nstr=skipper(nstr);
                                        indexed = nstr.indexOf(" ");
                                        oweight = nstr.substring(0,indexed);
                                        changer = "y";
                                }
                                sqltext = "insert into redistribute
values('"+RN+"','"+ospfnum+"','ospf','"+other_ospf+"','"+oweight+"','"+changer+"')";
                                state.executeUpdate(sqltext);
                        }
                }
                else if(nstr.startsWith("network")){ //OSPF information
                        nstr = skipper(nstr);
                        indexed = nstr.indexOf(" ");
                        ip = nstr.substring(0,indexed);
                        nstr = skipper(nstr);
                        indexed = nstr.indexOf(" ");
                        nm = hostmasker(nstr.substring(0,indexed));
                        ip+="/"+nm;
                        nstr = skipper(nstr);
                        nstr = skipper(nstr);
                        area = nstr;
                        nnstr = "insert into ospf
values('"+RN+"','"+ospfnum+"','"+ip+"','"+area+"','";
                        olist.add(nnstr);
                }
                else if(nstr.startsWith("distance")){ //OSPF weight information
                        nstr = skipper(nstr);
                        nstr = skipper(nstr);
                        if(nstr.startsWith("external")){
                                nstr = skipper(nstr);
                                exter = nstr;
                        }
                        else if(nstr.startsWith("inter-area")){
                                nstr = skipper(nstr);
                                intera = nstr;
                        }
                        else if(nstr.startsWith("intra-area")){
                                nstr = skipper(nstr);
                                intraa = nstr;
                        }

                }
                if((nstr = in.readLine()) == null)
                        break;
                else
                        nstr = nstr.trim();
        }
        for(int i=0;i<olist.size();i++){ //Insert information into the database
                sqltext = (String)olist.get(i) +
weight+"','"+exter+"','"+intera+"','"+intraa+"','internal')";
                state.executeUpdate(sqltext);
        }
        return nstr;
}

//Skip to the next word on the line
public String skipper(String str) throws Exception{
        int indexer = str.indexOf(" ")+1;
        String nstr = str.substring(indexer);
```

```
                return (nstr);
        }

        public void Connector(String ipAddress, String database, String user, String pass)
throws Exception{
                DatabaseMetaData dbmd;
                String url = "jdbc:postgresql://"+ipAddress+"/"+database;
                Properties props = new Properties();
                props.setProperty("user",user);
                props.setProperty("password",pass);
                db = DriverManager.getConnection(url,props);
                dbmd = db.getMetaData();
        }

      private String getInput(String promptText)
      {
            // open up standard input
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            String input = null;
            System.out.print(promptText);
            try {
               input = br.readLine();
            } catch (IOException ioe) {
               System.out.println("IO error!");
                      System.exit(1);
               }

               return input;
      }
}
```

APPENDIX B. DIJKSTRA.JAVA

```
/*
  * (C) Copyright 2005, by Scott Preston and Preston Research LLC
  *
  * Project Info:  http://www.javarobots.org
  *
  * This program is free software; you can redistribute it and/or
  * modify it under the terms of the GNU General Public License
  * as published by the Free Software Foundation; either version 2
  * of the License, or (at your option) any later version.
  *
  * This program is distributed in the hope that it will be useful,
  * but WITHOUT ANY WARRANTY; without even the implied warranty of
  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  * GNU General Public License for more details.
  *
  * You should have received a copy of the GNU General Public License
  * along with this program; if not, write to the Free Software
  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
  *
  */
//package com.scottpreston.javarobot.chapter7;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;

public class Dijkstra {

    private ArrayList vertices = new ArrayList();
    private ArrayList edges = new ArrayList();
    private HashMap oldVertex = new HashMap();
    private HashMap distances = new HashMap();
    private HashSet unsettled = new HashSet();
    private HashSet settled = new HashSet();

    public void addEdge(Edge e) {
        edges.add(e);
    }

    public void addAllEdges(ArrayList e) {
        edges = e;
    }

    public void addVertex(Vertex v) {
        vertices.add(v);
    }

    public void addAllVertices(ArrayList v) {
        vertices = v;
    }

    public int getDist(Vertex start, Vertex end) {
        int[][] adj = getAdj();
        int size = vertices.size();
        int w = 0;
        for (int i = 0; i < size; i++) {
            Vertex vi = (Vertex) vertices.get(i);
            for (int j = 0; j < size; j++) {
                Vertex vj = (Vertex) vertices.get(j);
                if (vi.equals(start) && vj.equals(end)) {
```

```
                w = adj[i][j];
            }
        }

    }
    return w;
}

public void setShortDistance(Vertex v, int dist) {
    unsettled.remove(v);
    distances.put(v, new Integer(dist));
    unsettled.add(v);
}

public void setPred(Vertex a, Vertex b ){
    oldVertex.put(a,b);
}

public Vertex getPred(Vertex a) {
    return (Vertex)oldVertex.get(a);
}

public int getShortDistance(Vertex v) {
    Integer d = (Integer) distances.get(v);
    if (d == null) {
        return Integer.MAX_VALUE;
    } else {
        return d.intValue();
    }
}

public Vertex extractMinimum() {
    Iterator i = unsettled.iterator();
    int min = Integer.MAX_VALUE;
    Vertex minV = null;
    while (i.hasNext()) {
        Vertex tmp = (Vertex) i.next();
        if (getShortDistance(tmp) < min) {
            min = getShortDistance(tmp);
            minV = tmp;
        }
    }
    unsettled.remove(minV);
    return minV;
}

public void relaxNeighbors(Vertex u) {
    int[][] adj = getAdj();
    int size = vertices.size();
    for (int i = 0; i < size; i++) {
        Vertex vi = (Vertex) vertices.get(i);
        if (vi.equals(u)) { // only check this i'th column
            for (int j = 0; j < size; j++) {
                Vertex v = (Vertex) vertices.get(j);
                int w2 = adj[i][j];
                // should give all adjacent vertices not settled
                if (w2 > 0 && w2 < Integer.MAX_VALUE
                        && (settled.contains(v) == false)) {
                    // sdoes a shorter distance exist?
                    if (getShortDistance(v) > getShortDistance(u)
                            + getDist(u, v)) {
                        int d = getShortDistance(u) + getDist(u, v);
                        setShortDistance(v, d);
                        setPred(v,u);

                    }
                }

            }
```

```
            }

        }
    }


    public Edge getShortestPath( Vertex start, Vertex end) {
        unsettled.add(start);
        setShortDistance(start,0);
       int badway = 0;
       Edge newer=null;
        while (unsettled.size() > 0) {
            Vertex u = extractMinimum(); // gets shortest Vertext
            settled.add(u);
            relaxNeighbors(u);
           if(badway == 1){
                  newer = new Edge(start,u,getDist(start,u));
           }
           badway++;
        }
        ArrayList<Vertex> l = new ArrayList<Vertex>();
        for (Vertex v = end; v != null; v = getPred(v)) {
            l.add(v);
        }
       int total = 0;
       Vertex v1,v2;
        Collections.reverse(l);
//        System.out.println("--- PRINT ORDER ---");
         v1 = (Vertex) l.get(0);
         for (int d=1;d < l.size();d++) {
            v2 = (Vertex) l.get(d);
           total += getDist(v1,v2);
           v1 = (Vertex) l.get(d);
  //          System.out.println(v.name);
        }
       Vertex ba,bb;
       ba = (Vertex) l.get(0);
        bb = (Vertex) l.get(1);
       newer = new Edge(ba,bb,getDist(ba,bb),total);
//     System.out.println(newer.v1.name+"\t"+newer.v2.name);
        return newer;
    }

    public Vertex getVertexByName(String n) {
        int size = vertices.size();
        for (int i = 0; i < size; i++) {
            Vertex vi = (Vertex) vertices.get(i);
            if (vi.name.equals(n)) {
                return vi;
            }
        }
        return null;

    }

    private int[][] getAdj() {

        int[][] adjMatrix = new int[vertices.size()][vertices.size()];
        // init all large
        for (int i = 0; i < vertices.size(); i++) {
            for (int j = 0; j < vertices.size(); j++) {
                adjMatrix[i][j] = Integer.MAX_VALUE;
            }
        }
        // set to actual values and selfs to zero
        for (int i = 0; i < vertices.size(); i++) {
            Vertex vi = (Vertex) vertices.get(i);
            for (int j = 0; j < vertices.size(); j++) {
                Vertex vj = (Vertex) vertices.get(j);
```

```
                if (i == j) {
                    adjMatrix[i][j] = 0;
                } else {
                    for (int k = 0; k < edges.size(); k++) {
                        Edge e = (Edge) edges.get(k);
                        if (e.v1.equals(vi) && e.v2.equals(vj))
                            adjMatrix[i][j] = e.weight;
                        //if (e.v2.equals(vi) && e.v1.equals(vj))
                          //  adjMatrix[i][j] = e.weight;
                    }
                }
            }
        }

        return adjMatrix;
    }


    public static void main(String[] args) {
    }

    /**
     * @return Returns the vertices.
     */
    public ArrayList getVertices() {
        return vertices;
    }
    /**
     * @param vertices The vertices to set.
     */
    public void setVertices(ArrayList vertices) {
        this.vertices = vertices;
    }
    /**
     * @return Returns the edges.
     */
    public ArrayList getEdges() {
        return edges;
    }
    /**
     * @param edges The edges to set.
     */
    public void setEdges(ArrayList edges) {
        this.edges = edges;
    }
}
```

APPENDIX C. EDGE.JAVA

```
/*
  * (C) Copyright 2005, by Scott Preston and Preston Research LLC
  *
  * Project Info:  http://www.javarobots.org
  *
  * This program is free software; you can redistribute it and/or
  * modify it under the terms of the GNU General Public License
  * as published by the Free Software Foundation; either version 2
  * of the License, or (at your option) any later version.
  *
  * This program is distributed in the hope that it will be useful,
  * but WITHOUT ANY WARRANTY; without even the implied warranty of
  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  * GNU General Public License for more details.
  *
  * You should have received a copy of the GNU General Public License
  * along with this program; if not, write to the Free Software
  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
  *
  */
//package com.scottpreston.javarobot.chapter7;

public class Edge {

    public String name;
    public Vertex v1;
    public Vertex v2;
    public int weight;
    public int total;

    public Edge() {}

    // constructs with two vertices and a weight
    public Edge(Vertex v1, Vertex v2, int w) {
        this.v1 = v1;
        this.v2 = v2;
        this.weight = w;
       this.total = 0;
    }

    // constructs with two vertices and a weight
    public Edge(Vertex v1, Vertex v2, int w, int t) {
        this.v1 = v1;
        this.v2 = v2;
        this.weight = w;
       this.total = t;
    }

//    public String toString() {
  //      return "{v1=" + v1.name + ",v2=" + v2.name + ",w=" + weight + "}";
  //  }

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. VERTEX.JAVA

```
/*
  * (C) Copyright 2005, by Scott Preston and Preston Research LLC
  *
  * Project Info:  http://www.javarobots.org
  *
  * This program is free software; you can redistribute it and/or
  * modify it under the terms of the GNU General Public License
  * as published by the Free Software Foundation; either version 2
  * of the License, or (at your option) any later version.
  *
  * This program is distributed in the hope that it will be useful,
  * but WITHOUT ANY WARRANTY; without even the implied warranty of
  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  * GNU General Public License for more details.
  *
  * You should have received a copy of the GNU General Public License
  * along with this program; if not, write to the Free Software
  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
  *
  */
//package com.scottpreston.javarobot.chapter7;

public class Vertex {

    public String name;

    public Vertex() {}

    public Vertex(String n) {
        name = n;
    }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. FINDROUTE.JAVA

```
//Stephen McManus
//Getting the routes
import java.sql.*;
import java.util.*;
import java.io.*;

public class FindRoute{
	private Connection db; //database connection
	private BufferedReader in; //Not probably used here but oh well
	private String RN; //Router name
	private String HN; //Host name
	 private Dijkstra dk; // Dijkstra structure
	private String sqltext; //sqltext
	private int rnum; //router number
	private Statement state; //inject sql queries
	private Statement state2; //inject sql queries
	private Statement state3; //inject sql queries
	private Statement state4; //inject sql queries
	private Statement stato; //inject sql queries
	private Statement stato2; //inject sql queries
	private Statement stato3; //inject sql queries
	private Statement statw; //inject sql queries
	private Statement statw2; //inject sql queries
	private Statement statw3; //inject sql queries
	private List<String> incIP; //IP without netmask bits
	private List<String> fullIP; //IP with netmask bits
	private List<String> netIP; //IP with netmask bits
	private List<String> incRouterName; //source router
	private List<String> outRouterName; //destination router
	private List<String> incInt; //source interface
	private List<String> outInt; //destination interface
	private List<String> troublerouters; //List of trouble routers

		//test.getROOspf(); //get redistributed ospf

	public static void main(String args[]) throws Exception{
		Class.forName("org.postgresql.Driver");
		FindRoute test = new FindRoute();
		test.Connector("172.20.106.36","mcmanst","mcmanst","fluffy"); //Connect to
the database
		int tester;
		test.getConnected(); //get connected routes
		test.getLoopback(); //get loopback routes
		test.getStatic(); //get static routes
		test.getRCOspf(); //get redistributed connected
		test.getRSOspf(); //get redistributed static
		test.getOspf();  //get ospf
		test.pruner(); //remove routes that are OSPF that are either connected or
static
		test.exex(); //label redistributed static and connected routes
		test.getROOspf(); //Run the algorithm
		test.setRed(); //Costs for redistributed routes
		test.getIntRedExt(); //Have routers that are internal find external routes
		test.getIntRedInt(); //Have routers that are internal rediscover internal
routes
		test.printOutput(); //print the show ip routes information
	}

	public void setRed() throws Exception{
		//Variables
		List<String> Rlist = new LinkedList<String>();
		List<String> Ilist = new LinkedList<String>();
		List<String> Ilist2 = new LinkedList<String>();
		List<String> Ilist3 = new LinkedList<String>();
```

```
		List<String> IPlist = new LinkedList<String>();
		List<Integer> Llist = new LinkedList<Integer>();
		List<String> Revlist = new LinkedList<String>();
		int total=0;
		int counter = 0;
		String area = null;
		String area2 = null;
		ResultSet rs,rs2;
		String rname=null;
		String intro = null;
		String intap = null;
		boolean breaker = false;
		boolean breaker2 = false;
		boolean done = false;

		//Gather redistributed OSPF routers and give them redistributed OSPF costs
(which is the maximum)
		sqltext = "select routername,routefrom,routeto from algorithm";
		rs = state.executeQuery(sqltext);
		while(rs.next()){
			intro = ""+Integer.MAX_VALUE;
			sqltext = "insert into morered
values('"+rs.getString(1)+"','"+rs.getString(2)+"','"+rs.getString(3)+"','"+intro+"')";
			state2.executeUpdate(sqltext);
		}
		//If there has been a redistributed metric set, then specify it in the
table, else leave it at the current state
		sqltext = "select routername,ospfinput,ospfoutput,cost from redistribute
where used = 'y' and protocol = 'ospf'";
		rs = state.executeQuery(sqltext);
		while(rs.next()){
			sqltext = "select distinct ospfarea from ospf where ospflabel =
'"+rs.getString(2)+"'";
			rs2 = state2.executeQuery(sqltext);
			if(rs2.next())
				area = rs2.getString(1);

			sqltext = "select cost from morered where routername !=
'"+rs.getString(1)+"' and routeto = '"+area+"'";
			rs2 = state2.executeQuery(sqltext);
			while(rs2.next()){
				total = Integer.parseInt(rs2.getString(1));
				counter = Integer.parseInt(rs.getString(4));
				//This is where it gets updated
				if(counter < total){
					sqltext = "update morered set cost =
'"+rs.getString(4)+"' where routername != '"+rs.getString(1)+"' and routeto =
'"+area+"'";
					state3.executeUpdate(sqltext);
				}
			}
		}

	}

	public void getIntRedExt() throws Exception{
		//Varibles
		List<String> Rlist = new LinkedList<String>();
		List<String> Ilist = new LinkedList<String>();
		List<String> Ilist2 = new LinkedList<String>();
		List<String> Ilist3 = new LinkedList<String>();
		List<String> IPlist = new LinkedList<String>();
		List<Integer> Llist = new LinkedList<Integer>();
		List<String> Revlist = new LinkedList<String>();
		int total=0;
		int counter = 0;
		int area = -1;
		int area2 = -1;
		ResultSet rs,rs2;
```

```
		String rname=null;
		String intro = null;
		String intap = null;
		boolean breaker = false;
		boolean breaker2 = false;
		boolean done = false;

		//Retreive the OSPF areas that are involved
		sqltext = "select ospfinput,ospfoutput from redistribute where protocol =
'ospf'";
		rs = state.executeQuery(sqltext);
		while(rs.next()){
			sqltext = "select distinct ospfarea from ospf where ospflabel =
'"+rs.getString(1)+"'";
			rs2 = state2.executeQuery(sqltext);
			if(rs2.next())
				if(Ilist.contains(rs2.getString(1))==false)
					Ilist.add(rs2.getString(1));


			sqltext = "select distinct ospfarea from ospf where ospflabel =
'"+rs.getString(2)+"'";
			rs2 = state2.executeQuery(sqltext);
			if(rs2.next())
				if(Ilist.contains(rs2.getString(1))==false)
					Ilist.add(rs2.getString(1));
		}
		//Go throught the areas in ascending order
		Collections.sort(Ilist);
		for(int i=0;i<Ilist.size();i++){
			for(int j=0;j<Ilist.size();j++){
				//If the areas do not match, meaning that the current
routes have been redistributed from another
				//area
				if(Ilist.get(j).compareTo(Ilist.get(i)) != 0){
					//Select the OSPF redistributed routers
					sqltext = "select distinct routername from
algorithm";
					rs = state.executeQuery(sqltext);
					while(rs.next())
						Rlist.add(rs.getString(1));
					//Obtain the network prefixes that do not pertain to
the routers OSPF area
					sqltext = "select distinct network(ip) from mainospf
where ospfarea = '"+Ilist.get(j)+"'";
					rs = state.executeQuery(sqltext);
					while(rs.next())
						Ilist2.add(rs.getString(1));
					//Obtain the routers from the OSPF area
					sqltext = "select distinct routername from
instancetab where instanceid = '"+Ilist.get(i)+"' except select distinct routername from
algorithm";
					rs = state.executeQuery(sqltext);
					while(rs.next())
						Ilist3.add(rs.getString(1));
					//While there are routers left in the area to
analyze
					while(Ilist3.isEmpty()==false){
						//Obtain the interface IP addresses that
pertain to the analyzed router
						for(int k=0;k<Rlist.size();k++){
							sqltext = "select network(ip) from
interface where routername = '"+Ilist3.get(0)+"' intersect select network(ip) from
interface where routername = '"+Rlist.get(k)+"'";
							rs = state.executeQuery(sqltext);
							while(rs.next()){
								sqltext = "select host(ip)
from interface where routername = '"+Rlist.get(k)+"' and network(ip) =
'"+rs.getString(1)+"'";
```

```
                                                        rs2 =
state2.executeQuery(sqltext);
                                                        while(rs2.next())

        IPlist.add(rs2.getString(1));
                                                }
                                        }
                                        //Grab the router's name
                                        rname = Ilist3.get(0);
                                        Ilist3.remove(0);
                                        //If there is no connected routes, put to
the end of the list, not connected
                                        //to routers that were just previously
analyzed
                                        if(IPlist.isEmpty()==true){
                                                Ilist3.add(Ilist3.size(),rname);
                                        }
                                        else{
                                                //Get the cost to read the
destination from the router
                                                //through interface and insert routes
that are lowest cost
                                                for(int k=0;k<Ilist2.size();k++){
                                                        for(int
l=0;l<IPlist.size();l++){

        Llist.add(getWeight(rname,Ilist2.get(k),IPlist.get(l)));
                                                        }
                                                        total = Integer.MAX_VALUE;
                                                        for(int
l=0;l<Llist.size();l++){
                                                                if(total>Llist.get(l))
                                                                        total =
Llist.get(l);
                                                        }
                                                        if(total !=
Integer.MAX_VALUE){
                                                                for(int
l=0;l<Llist.size();l++){

        if(Llist.get(l)==total){
                                                                                sqltext
= "select interfacename from interface where routername = '"+rname+"' and network(ip) >>=
'"+IPlist.get(l)+"'";
                                                                                rs =
state.executeQuery(sqltext);

        if(rs.next()){

        sqltext = "insert into output
values('"+rname+"','O','E2','"+Ilist2.get(k)+"','"+IPlist.get(l)+"','"+rs.getString(1)+"'
,'"+Ilist.get(i)+"')";

        state2.executeUpdate(sqltext);
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                        Llist.clear();
                                                }
                                                Rlist.clear();
                                                Rlist.add(rname);

                                        }
                                        IPlist.clear();
                                        Llist.clear();
                                }
                        }
                        Ilist2.clear();
```

```
                        Ilist3.clear();
                        Rlist.clear();
                }
        }
}

public void getIntRedInt() throws Exception{
        //Variables
        List<String> Rlist = new LinkedList<String>();
        List<String> Ilist = new LinkedList<String>();
        List<String> Ilist2 = new LinkedList<String>();
        List<String> Ilist3 = new LinkedList<String>();
        List<String> IPlist = new LinkedList<String>();
        List<String> IPlist2 = new LinkedList<String>();
        List<Integer> Llist = new LinkedList<Integer>();
        List<String> Revlist = new LinkedList<String>();
        int total=0;
        int counter = 0;
        int area = -1;
        int area2 = -1;
        ResultSet rs,rs2;
        String rname=null;
        String intro = null;
        String intap = null;
        boolean breaker = false;
        boolean breaker2 = false;
        boolean done = false;

        //Retreive the OSPF areas that are involved
        sqltext = "select ospfinput,ospfoutput from redistribute where protocol =
'ospf'";
        rs = state.executeQuery(sqltext);
        while(rs.next()){
                sqltext = "select distinct ospfarea from ospf where ospflabel =
'"+rs.getString(1)+"'";
                rs2 = state2.executeQuery(sqltext);
                if(rs2.next())
                        if(Ilist.contains(rs2.getString(1))==false)
                                Ilist.add(rs2.getString(1));


                sqltext = "select distinct ospfarea from ospf where ospflabel =
'"+rs.getString(2)+"'";
                rs2 = state2.executeQuery(sqltext);
                if(rs2.next())
                        if(Ilist.contains(rs2.getString(1))==false)
                                Ilist.add(rs2.getString(1));
        }
        //Go throught the areas in ascending order
        Collections.sort(Ilist);
        for(int i=0;i<Ilist.size();i++){
                for(int j=0;j<Ilist.size();j++){
                        //If the areas do not match, meaning that the current
routes have been redistributed from another
                        //area
                        if(Ilist.get(j).compareTo(Ilist.get(i)) == 0){
                                //Select the OSPF redistributed routers
                                sqltext = "select distinct routername from
algorithm";
                                rs = state.executeQuery(sqltext);
                                while(rs.next())
                                        Rlist.add(rs.getString(1));
                                //Obtain the routers from the OSPF area
                                sqltext = "select distinct routername from
instancetab where instanceid = '"+Ilist.get(i)+"' except select distinct routername from
algorithm";
                                rs = state.executeQuery(sqltext);
                                while(rs.next())
                                        Ilist3.add(rs.getString(1));
```

```
                                        //While there are routers left in the area to
analyze
                                        while(Ilist3.isEmpty()==false){
                                                //Obtain the interface IP addresses that
pertain to the analyzed router
                                                for(int k=0;k<Rlist.size();k++){
                                                        sqltext = "select network(ip) from
interface where routername = '"+Ilist3.get(0)+"' intersect select network(ip) from
interface where routername = '"+Rlist.get(k)+"'";
                                                        rs = state.executeQuery(sqltext);
                                                        while(rs.next()){
                                                                sqltext = "select host(ip)
from interface where routername = '"+Rlist.get(k)+"' and network(ip) =
'"+rs.getString(1)+"'";
                                                                rs2 =
state2.executeQuery(sqltext);
                                                                while(rs2.next())

        IPlist.add(rs2.getString(1));
                                                        }
                                                }
                                                //Grab the router's name
                                                rname = Ilist3.get(0);
                                                Ilist3.remove(0);
                                                //If there is no connected routes, put to
the end of the list, not connected
                                                //to routers that were just previously
analyzed
                                                if(IPlist.isEmpty()==true){
                                                        Ilist3.add(Ilist3.size(),rname);
                                                }
                                                else{
                                                        IPlist.clear();
                                                        //Obtain the interface IP addresses
that pertain to the analyzed router
                                                        sqltext = "select distinct ip from
ospf where routername = '"+rname+"'";
                                                        rs = state.executeQuery(sqltext);
                                                        while(rs.next())
                                                                IPlist2.add(rs.getString(1));
                                                        for(int y=0;y<IPlist2.size();y++){
                                                                sqltext = "select host(ip)
from interface where network(ip) = '"+IPlist2.get(y)+"' and routername != '"+rname+"'";
                                                                rs =
state.executeQuery(sqltext);
                                                                while(rs.next())

        IPlist.add(rs.getString(1));

                                                        }
                                                        IPlist2.clear();
                                                        //Obtain the network prefixes that do
pertain to the routers OSPF area

                                                        sqltext = "select distinct
network(ip) from mainospf where ospfarea = '"+Ilist.get(i)+"' except select distinct
network(ip) from mainospf where ospfarea = '"+Ilist.get(i)+"' and routername =
'"+rname+"'";
                                                        rs = state.executeQuery(sqltext);
                                                        while(rs.next())
                                                                Ilist2.add(rs.getString(1));
                                                        //Get the cost to read the
destination from the router
                                                        //through interface and insert routes
that are lowest cost
                                                        for(int k=0;k<Ilist2.size();k++){
                                                                for(int
l=0;l<IPlist.size();l++){
```

```
        Llist.add(getWeight(rname,Ilist2.get(k),IPlist.get(l)));
                                                                }
                                                                total = Integer.MAX_VALUE;
                                                                for(int
l=0;l<Llist.size();l++){
                                                                        if(total>Llist.get(l))
                                                                                total =
Llist.get(l);
                                                                }
                                                                if(total !=
Integer.MAX_VALUE){
                                                                        for(int
l=0;l<Llist.size();l++){

        if(Llist.get(l)==total){
                                                                                        String
intip = null;
                                                                                        sqltext
= "select interfaceip from output where routername = '"+rname+"' and ip =
'"+Ilist2.get(k)+"'";
                                                                                        rs =
state.executeQuery(sqltext);

        if(rs.next()){

        intip = rs.getString(1);
                                                                                        }
                                                                                        sqltext
= "select interfacename from interface where routername = '"+rname+"' and network(ip) >>=
'"+IPlist.get(l)+"'";
                                                                                        rs =
state.executeQuery(sqltext);

        if(rs.next()){

        if(intip.compareTo(IPlist.get(l))!=0){

        sqltext = "update output set interfaceip = '"+IPlist.get(l)+"', interface =
'"+rs.getString(1)+"', ospfredistribute = 'E2' where typeroute = 'O' and routername =
'"+rname+"' and ip = '"+Ilist2.get(k)+"'";

        state2.executeUpdate(sqltext);

        }
                                                                                        }
                                                                                }
                                                                        }
                                                                }
                                                                Llist.clear();
                                                        }
                                                        Rlist.clear();
                                                        Rlist.add(rname);

                                                }
                                                IPlist.clear();
                                                Llist.clear();
                                        }
                                }
                                Ilist2.clear();
                                Ilist3.clear();
                                Rlist.clear();
                        }
                }
        }

        public int getWeight(String name,String dest,String inter) throws Exception{
                //Varibles
                List<String> Rlist = new ArrayList<String>();
```

```
		List<String> Ilist = new ArrayList<String>();
		List<String> Ilist2 = new ArrayList<String>();
		List<String> IPlist = new ArrayList<String>();
		List<String> Llist = new ArrayList<String>();
		List<String> Revlist = new ArrayList<String>();
		int total=0;
		int outtotal=Integer.MAX_VALUE;
		int total2=Integer.MAX_VALUE;
		int counter = 0;
		int area = -1;
		int area2 = -1;
		int area4 = -1;
		ResultSet rw,rw2,rw3;
		String rname=null;
		String intro = null;
		String intap = null;
		String area3 = null;
		boolean breaker = false;
		boolean breaker2 = false;
		boolean done = false;
		//Get redistributed static and connected routes
		sqltext = "select distinct network(ip),ospfcost from tempospf order by
ospfcost asc";
		rw = statw.executeQuery(sqltext);
		while(rw.next()){
			Ilist.add(rw.getString(1));
			Ilist2.add(rw.getString(2));
		}
		//Get redistributed OSPF routers
		sqltext = "select distinct routername from algorithm";
		rw = statw.executeQuery(sqltext);
		while(rw.next())
			Revlist.add(rw.getString(1));


		//If redistributed OSPF routers and contains a redistributed connected and
static route
		if(Ilist.contains(dest)==true && breaker == false &&
Revlist.contains(name)==true){
			sqltext = "select distinct ospfarea from mainospf where ip
>>='"+inter+"'";
			rw = statw.executeQuery(sqltext);
			if(rw.next())
				area4 = Integer.parseInt(rw.getString(1));
			sqltext = "select distinct ospfarea from mainospf where ip =
'"+dest+"'";
			rw = statw.executeQuery(sqltext);
			if(rw.next()){
				area3 = rw.getString(1);
				sqltext = "select cost from morered where routername =
'"+name+"' and routeto = '"+area3+"'";
				rw2 = statw2.executeQuery(sqltext);
				total2 = Integer.MAX_VALUE;

				while(rw2.next()){
					//If the cost is specified and going into the area
of the redistributed route stop the algorithm
					if(Integer.parseInt(rw2.getString(1)) !=
Integer.MAX_VALUE && area4 == Integer.parseInt(area3)){
						breaker2 = true;
						done = true;
					}
					//If a lower cost, reset the cost
					if(total2 > Integer.parseInt(rw2.getString(1))){
						total2 = Integer.parseInt(rw2.getString(1));
					}
				}
			}
			//If this is the right cost, finish out the program
```

```
                        if(breaker2 == true){
                                total = total2;
                                return total;
                        }
                }


                //The cost and area of the interface ip of where you are starting from
                sqltext = "select ospfcost,ospfarea from ospf where network(ip) >>=
'"+inter+"' and routername = '"+name+"'";
                rw = statw.executeQuery(sqltext);
                if(rw.next()){
                        total += Integer.parseInt(rw.getString(1));
                        area = Integer.parseInt(rw.getString(2));
                }
                //The router of the first hop
                sqltext = "select routername from ospf where network(ip) >>= '"+inter+"'
and routername != '"+name+"'";
                rw = statw.executeQuery(sqltext);
                if(rw.next())
                        rname = rw.getString(1);
                //While there is no loop or found a route, breaker will be set to true
when found a route
                while(Rlist.contains(rname)==false && breaker == false){
                        Rlist.add(rname);
                        //Find the interface IP address and type of route of how to get to
the next link to the destination
                        sqltext = "select interfaceip,typeroute from output where
routername = '"+rname+"' and ip = '"+dest+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next()){
                                intap = rw.getString(1);
                                intro = rw.getString(2);
                        }
                        //Use the network IP address instead of the interface IP address
                        sqltext = "select network(ip) from interface where routername =
'"+rname+"' and ip >>= '"+intap+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next())
                                intap = rw.getString(1);
                        //If this is an OSPF redistributed router
                        if(Revlist.contains(rname)==true){
                                total = 0;
                                //Obtain OSPF label that have a specified redistributed
cost
                                sqltext = "select distinct ospfoutput from redistribute
where routername = '"+rname+"' and used = 'y'";
                                rw = statw.executeQuery(sqltext);
                                while(rw.next()){
                                        Llist.add(rw.getString(1));
                                }
                                //Go through those OSPF labels
                                for(int i=0;i<Llist.size();i++){
                                        sqltext = "select distinct network(ip) from mainospf
where routername = '"+rname+"' and ospflabel = '"+Llist.get(i)+"'";
                                        rw = statw.executeQuery(sqltext);
                                        IPlist.clear();
                                        while(rw.next()){
                                                IPlist.add(rw.getString(1));
                                        }
                                        //If this has a specified redistributed cost and
goes into the area you need, use that cost
                                        if(IPlist.contains(intap)==true){
                                                sqltext = "select distinct ospfarea from
ospf where ospflabel = '"+Llist.get(i)+"' and routername = '"+rname+"'";
                                                rw = statw.executeQuery(sqltext);

                                                if(rw.next())
```

```
                                                area2 =
Integer.parseInt(rw.getString(1));
                                        if(area==area2)
                                                breaker = true;
                                        else{
                                                area = area2;

                                                sqltext = "select cost from
redistribute where routername = '"+rname+"' and ospfoutput = '"+Llist.get(i)+"' and
protocol = 'ospf'";

                                                rw2 = statw2.executeQuery(sqltext);
                                                if(rw2.next()){

                                                        breaker = true;
                                                        breaker2 = true;
                                                        done = true;
                                                        total =
Integer.parseInt(rw2.getString(1));
                                                }
                                                break;
                                        }
                                }
                        }
                        Llist.clear();
                        //if have gone back to the same area, this has caused the
loop and done
                        sqltext = "select ospfarea from ospf where routername =
'"+rname+"' and ip = '"+intap+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next())
                                if(area == Integer.parseInt(rw.getString(1)))
                                        break;
                        //if destination is static or connected redistributed route
                        if(Ilist.contains(dest)==true && breaker == false){
                                sqltext = "select distinct ospfarea from mainospf
where ip >>='"+intap+"'";
                                rw = statw.executeQuery(sqltext);
                                if(rw.next())
                                        area4 = Integer.parseInt(rw.getString(1));
                                sqltext = "select distinct ospfarea from mainospf
where ip = '"+dest+"'";
                                rw = statw.executeQuery(sqltext);
                                if(rw.next()){
                                        area3 = rw.getString(1);
                                        sqltext = "select cost from morered where
routername = '"+rname+"' and routeto = '"+area3+"'";
                                        rw2 = statw2.executeQuery(sqltext);
                                        total2 = Integer.MAX_VALUE;
                                        while(rw2.next()){
                                                //If the cost is specified and going
into the area of the redistributed
                                                //route stop the algorithm
                                                if(Integer.parseInt(rw2.getString(1))
!= Integer.MAX_VALUE && area4 == Integer.parseInt(area3)){
                                                        breaker2 = true;
                                                        done = true;
                                                }
                                                //If a lower cost, reset the cost
                                                if(total2 >
Integer.parseInt(rw2.getString(1)))
                                                        total2 =
Integer.parseInt(rw2.getString(1));
                                        }
                                }
                                //If this is the right cost, finish out the program
                                if(breaker2 == true){
                                        total = total2;
                                        break;
```

```
                                }
                        }

                }
                //If connected or static, finished and add the last cost and return
that as the cost
                if((intro.compareTo("C")==0 || intro.compareTo("S")==0) && breaker
== false){
                        breaker = true;
                        done = true;
                        sqltext = "select ospfcost from mainospf where network(ip)
>>= '"+intap+"' and routername = '"+rname+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next()){
                                total += Integer.parseInt(rw.getString(1));
                        }

                }
                //If just another OSPF router, keep going through the algorithm
                if((intro.compareTo("C")!=0 && intro.compareTo("S")!=0) && breaker
== false){
                        sqltext = "select ospfcost from ospf where network(ip) >>=
'"+intap+"' and routername = '"+rname+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next())
                                total += Integer.parseInt(rw.getString(1));
                        sqltext = "select routername from ospf where network(ip)
>>= '"+intap+"' and routername != '"+rname+"'";
                        rw = statw.executeQuery(sqltext);
                        if(rw.next())
                                rname = rw.getString(1);
                        }
                }
                //If there is a loop
                if(done == false)
                        total = Integer.MAX_VALUE;
                else{
                        //If a static or connected redistributed route, use the lowest cost
of that route
                        if(Ilist.contains(dest)==true && breaker2 == false){
                                int indexer;
                                indexer = Ilist.indexOf(dest);
                                total = Integer.parseInt(Ilist2.get(indexer));
                        }
                }
                return total;
        }

        public void getConnected() throws Exception{
                incIP=new ArrayList<String>();
                incRouterName=new ArrayList<String>();
                fullIP=new ArrayList<String>();
                netIP=new ArrayList<String>();
                outRouterName=new ArrayList<String>();
                incInt=new ArrayList<String>();
                outInt=new ArrayList<String>();

                String sqltext;
                List<String> lister = new ArrayList<String>(); //Unique ip address
                List<String> IP = new ArrayList<String>();//IP
                List<String> RouterName = new ArrayList<String>(); //Routername
                List<String> hostIP = new ArrayList<String>(); //IP without netmask bits
                List<String> IntName = new ArrayList<String>(); //interface name


                ResultSet rs;
                sqltext = "select distinct network(ip) from interface";
                rs = state.executeQuery(sqltext);
                while(rs.next()){ //Collect the unique ip address
```

```
                        lister.add(rs.getString(1));
                }
                for(int i=0;i<lister.size();i++){
                        sqltext = "select routername,ip,host(ip),interfacename from
interface where ip <<='" + lister.get(i) + "'";
                        rs = state.executeQuery(sqltext);
                        while(rs.next()){ //Get the number of ip addresses that have
                                RouterName.add(rs.getString(1));
                                IP.add(rs.getString(2));
                                hostIP.add(rs.getString(3));
                                IntName.add(rs.getString(4));
                        }
                        if(RouterName.size() ==2) //If two are connected to each other show
that
                        {
                                incIP.add(hostIP.get(0));
                                incRouterName.add(RouterName.get(0));
                                fullIP.add(IP.get(0));
                                outRouterName.add(RouterName.get(1));
                                incInt.add(IntName.get(0));
                                outInt.add(IntName.get(1));
                                netIP.add(lister.get(i));
                                incIP.add(hostIP.get(1));
                                incRouterName.add(RouterName.get(1));
                                fullIP.add(IP.get(1));
                                outRouterName.add(RouterName.get(0));
                                incInt.add(IntName.get(1));
                                outInt.add(IntName.get(0));
                                netIP.add(lister.get(i));
                        }

                        RouterName.clear();
                        IP.clear();
                        hostIP.clear();
                        IntName.clear();
                }
        }

        public void getOspf() throws Exception{
                //Unique items
                troublerouters=new ArrayList<String>();
                List<String> unrn = new ArrayList<String>(); //Unique Routername
                List<String> unrn2 = new ArrayList<String>(); //Unique Routername
                List<String> unip = new ArrayList<String>(); //Unique IP address
                List<String> lunip = new ArrayList<String>(); //Used not to include
specific connected ip address ranges
                List<String> notip = new ArrayList<String>(); //Used to not include
directly connected ip address ranges

                String sqltext;
                //Items gathered from the database and used shortly
                List<String> lister = new ArrayList<String>(); //List of unique areas
                List<String> lister2 = new ArrayList<String>(); //List of unique ip
addresses from OSPF
                List<String> ospf_rn = new ArrayList<String>(); //OSPF router name
                List<String> ospf_ospfnum = new ArrayList<String>(); //OSPF area number
                List<String> ospf_ip = new ArrayList<String>(); //OSPF ip addresses
                List<String> ospf_weight = new ArrayList<String>(); //OSPF weight
                List<String> int_intn = new ArrayList<String>(); //Interface interface
name
                List<String> int_ext = new ArrayList<String>(); //External distance
                List<String> int_era = new ArrayList<String>(); //Internal distance
                List<String> int_aar = new ArrayList<String>(); //Intraarea distance
                List<String> int_type = new ArrayList<String>(); //What type of route
                List<String> int_ip = new ArrayList<String>(); //Interface IP address


                //Connected graph information
                List<String> inRN = new ArrayList<String>(); //Start router
```

```
		List<String> ouRN = new ArrayList<String>(); //End router
		List<String> inIP = new ArrayList<String>(); //Start IP address
		List<String> ouIP = new ArrayList<String>(); //End IP address
		List<String> fulIP = new ArrayList<String>(); //Network IP address
		List<String> inint = new ArrayList<String>(); //Start Interface
		List<String> outint = new ArrayList<String>(); //End Interface
		List<String> weight = new ArrayList<String>(); //Cost of the network
		List<String> externals = new ArrayList<String>(); //External preference
		List<String> interareas = new ArrayList<String>(); //Internal preference
		List<String> intraareas = new ArrayList<String>(); //Intraarea preference
		List<String> typers = new ArrayList<String>(); //What type of route


		ResultSet rs,rs2;
		sqltext = "select distinct ospfarea from mainospf";
		rs = state.executeQuery(sqltext);
		while(rs.next()){ //Collect the unique areas
			lister.add(rs.getString(1));
		}
		for(int j=0;j<lister.size();j++){ //This is per area
			sqltext = "select distinct interface.ip from ospf join interface on
ospf.ip >>= interface.ip and ospf.routername = interface.routername where ospf.ospfarea =
'" + lister.get(j)+"'";
			rs = state.executeQuery(sqltext);
			while(rs.next()){ //Collect the unique ip address from the
interface table to determine the connection
				unip.add(rs.getString(1));
			}
			sqltext = "select distinct ospf.routername from ospf join interface
on ospf.ip >>= interface.ip and ospf.routername = interface.routername where
ospf.ospfarea = '" + lister.get(j)+"'";
			rs = state.executeQuery(sqltext);
			while(rs.next()){ //Collect the unique router names
				unrn.add(rs.getString(1));
			}
			sqltext = "select distinct routername from mainospf where ospfarea
= '" + lister.get(j)+"'";
			rs = state.executeQuery(sqltext);
			while(rs.next()){ //Collect the unique router names
				unrn2.add(rs.getString(1));
			}
			sqltext = "select distinct ospf.ip from ospf join interface on
ospf.ip >>= interface.ip and ospf.routername = interface.routername where ospf.ospfarea =
'" + lister.get(j)+"'";
			rs = state.executeQuery(sqltext);
			while(rs.next()){ //Collect the unique ip address from the ospf
table to determine the number of connections
				lister2.add(rs.getString(1));
			}
			for(int i=0;i<lister2.size();i++){
				// This is only for internal routes
				sqltext = "select
ospf.routername,ospf.ospflabel,ospf.ip,ospf.ospfcost,interface.interfacename,interface.ip
,ospf.externalad,ospf.interareaad,ospf.intraareaad,ospf.typeredistribute from ospf join
interface on ospf.ip >>= interface.ip and ospf.routername = interface.routername where
ospfarea = '" + lister.get(j) + "'and ospf.ip <<= '"+lister2.get(i)+"' and
typeredistribute = 'internal' order by ospf.routername";
				rs = state.executeQuery(sqltext);
				//Get the info for the connected routers
				while(rs.next()){
					ospf_rn.add(rs.getString(1));
					ospf_ospfnum.add(rs.getString(2));
					ospf_ip.add(rs.getString(3));
					ospf_weight.add(rs.getString(4));
					int_intn.add(rs.getString(5));
					int_ip.add(rs.getString(6));
					int_ext.add(rs.getString(7));
					int_era.add(rs.getString(8));
					int_aar.add(rs.getString(9));
```

```
                                int_type.add(rs.getString(10));
                        }
                        //Make the table of connected nodes
                        if(ospf_rn.size()==2){
                                inRN.add(ospf_rn.get(0));
                                ouRN.add(ospf_rn.get(1));
                                inIP.add(int_ip.get(0));
                                ouIP.add(int_ip.get(1));
                                fulIP.add(ospf_ip.get(0));
                                inint.add(int_intn.get(0));
                                outint.add(int_intn.get(1));
                                weight.add(ospf_weight.get(0));
                                externals.add(int_ext.get(0));
                                interareas.add(int_era.get(0));
                                intraareas.add(int_aar.get(0));
                                typers.add(int_type.get(0));
                                inRN.add(ospf_rn.get(1));
                                ouRN.add(ospf_rn.get(0));
                                inIP.add(int_ip.get(1));
                                ouIP.add(int_ip.get(0));
                                fulIP.add(ospf_ip.get(1));
                                inint.add(int_intn.get(1));
                                outint.add(int_intn.get(0));
                                weight.add(ospf_weight.get(1));
                                externals.add(int_ext.get(1));
                                interareas.add(int_era.get(1));
                                intraareas.add(int_aar.get(1));
                                typers.add(int_type.get(1));

                        }
                        //Clear out everything
                        ospf_rn.clear();
                        ospf_ospfnum.clear();
                        ospf_ip.clear();
                        ospf_weight.clear();
                        int_intn.clear();
                        int_ip.clear();
                        int_ext.clear();
                        int_era.clear();
                        int_aar.clear();
                        int_type.clear();
                }

                //ExperiMENTAL code
                sqltext = "select distinct text(ip) from tempospf where ospfarea =
'"+lister.get(j)+"'";
                rs = state.executeQuery(sqltext);
                while(rs.next()){
                        unip.add(rs.getString(1));
                }
                String maker = "nainterface";
                String maker2 = "0";
                String maker3 = "internal";
                sqltext = "select
routername,ospflabel,network(ip),ospfarea,ospfcost from tempospf where ospfarea =
'"+lister.get(j)+"' and (routername like '%l%' or routername like '%s%')";
                rs = state.executeQuery(sqltext);
                while(rs.next()){
                        sqltext = "select routername from tempospf where ospfarea =
'"+rs.getString(4)+"' and ospflabel = '"+rs.getString(2)+"' and network(ip) =
'"+rs.getString(3)+"' and ospfcost = '"+rs.getString(5)+"' and routername !=
'"+rs.getString(1)+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                inRN.add(rs2.getString(1));
                                ouRN.add(rs.getString(1));
                                inIP.add(rs.getString(3));
                                ouIP.add(rs.getString(3));
                                fulIP.add(rs.getString(3));
```

```
                                        inint.add(maker);
                                        outint.add(maker);
                                        weight.add(rs.getString(5));
                                        externals.add(maker2);
                                        interareas.add(maker2);
                                        intraareas.add(maker2);
                                        typers.add(maker3);
                                }
                        }
                        for(int k=0;k<unrn.size();k++){
                                // Clearing out everything and setting variables
                                lunip.clear();
                                String vn;
                                int en,en2,en3,ind,ded;
                                String bunip,namer;
                                int munip,lunar;
                                notip.clear();


                                List<Vertex> lvert= new LinkedList<Vertex>(); //List of
vertices
                                List<String> lrn = new ArrayList<String>(); //Unique area
                                List<String> lint = new ArrayList<String>(); //Unique area
                                List<String> usedip = new ArrayList<String>(); //IP
addresses that have already been looked and no need to repeat them

                                Vertex a;

                                //Go through the table of information and grab the network
ip addresses needed
                                for(int q=0;q<inRN.size();q++){
                                        if(inRN.get(q).compareTo(unrn.get(k))==0){
                                                notip.add(fulIP.get(q));
                                        }
                                }

                                //Go through the table and grab the from and to ip
addresses used to exclude looking at directly connected links
                                for(int l=0;l<inRN.size();l++){
                                        if(notip.contains(fulIP.get(l))){
                                                lunip.add(inIP.get(l));
                                                lunip.add(ouIP.get(l));
                                        }
                                }
                                //Go through the ip address range
                                for(int l=0;l<unip.size();l++){
                                        int bind,clues;
                                        bind = inIP.indexOf(unip.get(l));// used to get an
index of the unique ip address
                                        //Go ahead and run Dijkstra if the ip address ranged
has been used or directly connected
                                        if(lunip.contains(unip.get(l))==false &&
usedip.contains(fulIP.get(bind)) == false){
                                                // make a new Dijkstra element
                                                dk = new Dijkstra();
                                                //Make all of the routers in this area a
vertex
                                                for(int n=0;n<unrn2.size();n++){
                                                        a = new Vertex(unrn2.get(n));
                                                        lvert.add(a);
                                                }

                                                //Add these vertices into Dijkstra class
                                                for(int n=0;n<lvert.size();n++){
                                                        dk.addVertex(lvert.get(n));
                                                }
                                                for(int n=0;n<inRN.size();n++){
                                                        ind =
unrn2.indexOf(inRN.get(n));//index of input router vertex
```

```
                                                ded =
unrn2.indexOf(ouRN.get(n));//index of output router vertex
                                                en =
Integer.parseInt(weight.get(n));//weight between the two vertices
                                                en2 =
Integer.parseInt(externals.get(n)); //external preference
                                                en3 =
Integer.parseInt(intraareas.get(n)); //intraarea preference


                                                        //Add the edges in the Dijkstra
algorithm
                                                        dk.addEdge(new
Edge(lvert.get(ind),lvert.get(ded),en));
                                        }
                                        int mindx,mindx2;
                                        String dind,dind2;
                                        //Get the router name of the unique ip
address to start with
                                        mindx = inIP.indexOf(unip.get(l));
                                        dind = inRN.get(mindx);
                                        dind2 = ouRN.get(mindx);
                                        Edge evert,evert2;

                                        //As long as the starting point is not
ending point, go ahead and do dijkstra
                                        if(dind.compareTo(unrn.get(k))!=0){
                                                String lyip,lyint,lyweight;
                                                int badder;
                                                int lytot,lytot2;
                                                //Run Dijkstra's algorithm
                                                evert =
dk.getShortestPath(lvert.get(unrn2.indexOf(unrn.get(k))),lvert.get(unrn2.indexOf(dind)));
                                                lytot = evert.total; //Cost of the
first hop to be able to find the correct interface
                                                evert2 =
dk.getShortestPath(lvert.get(unrn2.indexOf(unrn.get(k))),lvert.get(unrn2.indexOf(dind2)))
;
                                                lytot2 = evert2.total; //Cost of the
first hop to be able to find the correct interface
                                                if(lytot < lytot2)
                                                        lyweight = ""+evert.weight;
                                                else
                                                        lyweight = ""+evert2.weight;
                                                badder = inIP.indexOf(unip.get(l));
//used to make sure to avoid duplicates

                                                //Find the correct occurance of the
link
                                                for(int n=0;n<inRN.size();n++){

        if(inRN.get(n).compareTo(evert.v1.name)==0)

        if(ouRN.get(n).compareTo(evert.v2.name)==0){

        if(weight.get(n).compareTo(lyweight)==0){
                                                                                sqltext
= "insert into output values
('"+unrn.get(k)+"','O','NULL','"+fulIP.get(badder)+"','"+ouIP.get(n).substring(0,ouIP.get
(n).indexOf("/"))+"','"+inint.get(n)+"','"+lister.get(j)+"')";

        state.executeUpdate(sqltext);

        usedip.add(fulIP.get(badder));
                                                                                break;
                                                                        }
                                                                }
                                                }
                                        }
```

```
                                }
                        lvert.clear();

                        }
                        usedip.clear();

                }
                //Clear everything out for the next run
                inRN.clear();
                ouRN.clear();
                inIP.clear();
                ouIP.clear();
                fulIP.clear();
                inint.clear();
                outint.clear();
                weight.clear();
                externals.clear();
                interareas.clear();
                intraareas.clear();
                typers.clear();
                unip.clear();
                unrn.clear();
                unrn2.clear();
                lister2.clear();

        }
    }

    public void getLoopback() throws Exception{
            String sqltext,badnet;
            badnet = "0.0.0.0/0";
            ResultSet rs;
            sqltext = "select routername,ip,host(ip),interfacename,network(ip) from
interface where lower(interfacename) like 'loop%'";
            rs = state.executeQuery(sqltext);
            while(rs.next()){ //Loopbacks that are connected
                    incIP.add(rs.getString(3));
                    incRouterName.add(rs.getString(1));
                    outRouterName.add(rs.getString(1));
                    fullIP.add(rs.getString(2));
                    incInt.add(rs.getString(4));
                    outInt.add(rs.getString(4));
                    netIP.add(rs.getString(5));
            }
            for(int i=0;i<fullIP.size();i++){
                    sqltext = "insert into output values
('"+incRouterName.get(i)+"','C','NULL','"+netIP.get(i)+"','"+netIP.get(i)+"','"+incInt.ge
t(i)+"','NULL')";
                    state.executeUpdate(sqltext);
            }
    }

    public void getStatic() throws Exception{
            List<String> name = new ArrayList<String>(); //Routername
            List<String> outip = new ArrayList<String>(); //Which ip address to try
and contact
            List<String> netip = new ArrayList<String>(); //Which ip address to try
and contact


            String sqltext,routerName,tempered;
            int indexer;
            ResultSet rs;
            sqltext = "select routername,host(interfaceip),network(ip) from static";
            rs = state.executeQuery(sqltext); //Collect all of the static routes
            while(rs.next()){
                    outip.add(rs.getString(2));
                    name.add(rs.getString(1));
                    netip.add(rs.getString(3));
```

```
                }
                for(int i=0; i<outip.size();i++)
                {
                        sqltext = "insert into output values
('"+name.get(i)+"','S','NULL','"+netip.get(i)+"','"+outip.get(i)+"','NULL','NULL')";
                        state.executeUpdate(sqltext);

                }
        }

        public void getRCOspf() throws Exception{
                String sqltext,routered,routered2,osnum,we,are,ext,iarea,iiarea,neip;
//Variables
                ResultSet rs,rs2,rs3; //Result sets used for the database
                sqltext = "select routername,ospfinput,cost from redistribute where
protocol = 'connected'";
                rs = state.executeQuery(sqltext);
                List<String> connectIP = new ArrayList<String>();
                are = "-1";
                while(rs.next()){ //Connected routes
                        int loopindex = 0;
                        routered = rs.getString(1); //Router name
                        osnum = rs.getString(2); //OSPF label number
                        we = rs.getString(3); //Weight
                        sqltext = "select ospfarea,network(ip) from ospf where routername =
'"+routered+"' and ospflabel = '"+osnum+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                are = rs2.getString(1); //OSPF area number
                                connectIP.add(rs2.getString(2));
                        }
                        ext = "0";
                        iarea = "0";
                        iiarea = "0";
                        sqltext = "select network(ip) from interface where routername =
'"+routered+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                if(connectIP.contains(rs2.getString(1)) == false){
                                        neip = rs2.getString(1); //Network IP address
                                        // Insert these routes into the ospf table as normal
routes
                                        sqltext = "insert into tempospf values
('"+routered+"','"+osnum+"','"+neip+"','"+are+"','"+we+"','"+ext+"','"+iarea+"','"+iiarea
+"','internal')";
                                        routered2 = routered + "l" + loopindex;
                                        state3.executeUpdate(sqltext);
                                        sqltext = "insert into tempospf values
('"+routered2+"','"+osnum+"','"+neip+"','"+are+"','"+we+"','"+ext+"','"+iarea+"','"+iiare
a+"','internal')";

                                        state3.executeUpdate(sqltext);
                                        loopindex++;
                                }
                        }
                }
        }

        public void getRSOspf() throws Exception{
                String sqltext,routered,routered2,osnum,we,are,ext,iarea,iiarea,neip;
//Variables
                ResultSet rs,rs2,rs3; //Result sets used for the database
                sqltext = "select routername,ospfinput,cost from redistribute where
protocol = 'static'";
                rs = state.executeQuery(sqltext);
                int loopindex = 0;
                are = "-1";
                while(rs.next()){ //Static routes
                        routered = rs.getString(1); //Router name
```

```
                        osnum = rs.getString(2); //OSPF label number
                        we = rs.getString(3); //Weight
                        sqltext = "select distinct ospfarea from ospf where routername =
'"+routered+"' and ospflabel = '"+osnum+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next())
                                are = rs2.getString(1);
                        ext = "0";
                        iarea = "0";
                        iiarea = "0";
                        sqltext = "select network(ip) from static where routername =
'"+routered+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                neip = rs2.getString(1); //Network IP address to reach
                                sqltext = "insert into tempospf values
('"+routered+"','"+osnum+"','"+neip+"','"+are+"','"+we+"','"+ext+"','"+iarea+"','"+iiarea
+"','internal')";
                                state3.executeUpdate(sqltext);
                                routered2 = routered + "s" + loopindex;
                                sqltext = "insert into tempospf values
('"+routered2+"','"+osnum+"','"+neip+"','"+are+"','"+we+"','"+ext+"','"+iarea+"','"+iiare
a+"','internal')";
                                state3.executeUpdate(sqltext);
                                loopindex++;

                        }
                }
        }

        public void getROOspf() throws Exception{
                //Variables
                String sqltext;
                String rern = null;
                String reria = null;
                String reroa = null;
                String reia = null;
                String reoa = null;
                List<String> areaList = new ArrayList<String>();
                List<String> sareaList = new ArrayList<String>();
                List<String> CList = new ArrayList<String>();
                List<String> RList = new ArrayList<String>();
                List<String> DList = new ArrayList<String>();
                List<String> IList = new ArrayList<String>();
                List<String> NList = new ArrayList<String>();
                ResultSet rs,rs2,rs3,rs4;
                List<String> er= new ArrayList<String>();
                List<String> es= new ArrayList<String>();
                List<String> ee= new ArrayList<String>();
                List<String> ds= new ArrayList<String>();
                List<String> dd= new ArrayList<String>();
                String focus = null;
                String tr = null;
                String ts = null;
                String te = null;
                int fcounter,scounter,fcom,scom;
                //Get the OSPF areas that are active in this redistribution
                sqltext = "select ospfinput,ospfoutput from redistribute where protocol =
'ospf'";
                rs = state.executeQuery(sqltext);
                while(rs.next()){
                        sqltext = "select routername,ospfarea,externalad from ospf where
ospflabel = '"+rs.getString(1)+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                rern = rs2.getString(1);
                                reria = rs2.getString(2);
                                reia = rs2.getString(3);
                        }
```

```
                        sqltext = "select ospfarea,externalad from ospf where ospflabel =
'"+rs.getString(2)+"'";
                        rs2 = state2.executeQuery(sqltext);
                        while(rs2.next()){
                                reroa = rs2.getString(1);
                                reoa = rs2.getString(2);
                        }
                        if(areaList.contains(reria) == false)
                                areaList.add(reria);
                        if(areaList.contains(reroa) == false)
                                areaList.add(reroa);
                        if(RList.contains(rern) == false)
                                RList.add(rern);
                        sqltext = "insert into algorithm
values('"+rern+"','"+reria+"','"+reroa+"','"+reia+"','n')";
                        state2.executeUpdate(sqltext);
                }
                //Insert information into the ritable table
                for(int i=0;i<RList.size();i++){
                        sqltext = "select distinct
ospf.routername,ospf.ospfarea,interface.interfacename,host(interface.ip) from ospf join
interface on interface.ip <<= ospf.ip and interface.routername = ospf.routername where
ospf.routername = '"+RList.get(i)+"'";
                        rs = state.executeQuery(sqltext);
                        while(rs.next()){
                                sqltext = "insert into ritable
values('"+rs.getString(1)+"','"+rs.getString(2)+"','"+rs.getString(3)+"','"+rs.getString(
4)+"')";
                                state2.executeUpdate(sqltext);
                        }
                }
                //Go throught the OSPF areas as a network prefix
                Collections.sort(areaList);
                for(int i=0;i<areaList.size();i++){
                        sareaList.add(areaList.get(i));
                        //Create the initial candidate list
                        sqltext = "select routername from algorithm where routefrom =
'"+areaList.get(i)+"'";
                        rs = state.executeQuery(sqltext);
                        while(rs.next())
                                CList.add(rs.getString(1));

                        int q=0;
                        //Making this random
                        Random chooser = new Random();
                        int u=0;
                        //while the candidate list is not empty
                        while(CList.isEmpty()==false){
                                u = chooser.nextInt(CList.size());
                                focus = CList.remove(u);
                                boolean breaker = false;
                                //Select inactive routes
                                sqltext = "select routefrom,routeto from algorithm where
active = 'n' and routername = '"+focus+"' order by ad asc";
                                rs = state.executeQuery(sqltext);
                                while(rs.next()){
                                        //While you have not found a route
                                        while(breaker == false){
                                                fcounter = 0;
                                                scounter = 0;
                                                fcom = -1;
                                                scom = -1;
                                                boolean tester = false;
                                                boolean tester2 = false;

                                                //If you are not at the start node and
information is not coming, pick a different path
                                                sqltext = "select routername from algorithm
where active = 'y' and routername != '"+focus+"' and routeto = '"+rs.getString(1)+"'";
```

```
                                        rs2 = state2.executeQuery(sqltext);
                                        while(rs2.next()){
                                                tester2 = true;
                                        }

        if(rs.getString(1).compareTo(areaList.get(i)) != 0){
                                                if(tester2 == false){
                                                        break;
                                                }
                                        }

                                        if(sareaList.contains(rs.getString(1)) ==
true)
                                                fcounter++;
                                        if(sareaList.contains(rs.getString(2)) ==
true)
                                                scounter++;
                                        //If information is coming from both areas,
pick the lowest
                                        if(fcounter == 1 && scounter == 1){
                                                sqltext = "select ad from algorithm
where routername = '"+focus+"' and routefrom = '"+rs.getString(1)+"' and routeto =
'"+rs.getString(2)+"'";
                                                rs2 = state2.executeQuery(sqltext);
                                                while(rs2.next())
                                                        fcom =
Integer.parseInt(rs2.getString(1));

                                                sqltext = "select ad from algorithm
where routername = '"+focus+"' and routefrom = '"+rs.getString(2)+"' and routeto =
'"+rs.getString(1)+"'";
                                                rs2 = state2.executeQuery(sqltext);
                                                while(rs2.next())
                                                        scom =
Integer.parseInt(rs2.getString(1));

                                                if(scom < 0){
                                                        tr = focus;
                                                        ts = rs.getString(1);
                                                        te = rs.getString(2);
                                                }
                                                else{
                                                        if(fcom < scom){
                                                                tr = focus;
                                                                ts = rs.getString(1);
                                                                te = rs.getString(2);
                                                        }
                                                        else{
                                                                tr = focus;
                                                                ts = rs.getString(2);
                                                                te = rs.getString(1);
                                                        }
                                                }
                                        }
                                        //If information only coming from the route
from
                                        else if(fcounter == 1 && scounter == 0){
                                                tr = focus;
                                                ts = rs.getString(1);
                                                te = rs.getString(2);
                                        }
                                        //If informatin only coming from the route
to
                                        else if(fcounter == 0 && scounter == 1){
                                                sqltext = "select ad from algorithm
where routername = '"+focus+"' and routefrom = '"+rs.getString(2)+"' and routeto =
'"+rs.getString(1)+"'";
                                                rs2 = state2.executeQuery(sqltext);
                                                while(rs2.next())
```

```
                                                        scom =
Integer.parseInt(rs2.getString(1));
                                                if(scom < 0)
                                                        break;
                                                else{
                                                        tr = focus;
                                                        ts = rs.getString(2);
                                                        te = rs.getString(1);
                                                }
                                        }
                                        else
                                                break;
                                        //If this route already exists, don't do
anything and just exit
                                        sqltext = "select routername from algorithm
where active = 'y' and routername = '"+tr+"' and routefrom = '"+ts+"' and routeto =
'"+te+"'";
                                        rs2 = state2.executeQuery(sqltext);
                                        if(rs2.next()){
                                                break;
                                        }
                                        sqltext = "select * from algorithm where
active = 'y' and routername = '"+tr+"' and routeto = '"+te+"' and routefrom = '"+ts+"'";
                                        rs2 = state2.executeQuery(sqltext);
                                        while(rs2.next())
                                                tester = true;
                                        //update the route when it has not been
inserted
                                        if(tester == false){
                                                breaker = true;
                                                sqltext = "select routeto from
algorithm where routername = '"+tr+"' and routeto != '"+te+"' and routefrom != '"+ts+"'
and active = 'y'";
                                                rs2 = state2.executeQuery(sqltext);
                                                while(rs2.next())
                                                        DList.add(rs2.getString(1));
                                                sqltext = "update algorithm set
active = 'n' where routername = '"+tr+"' and routeto != '"+te+"' and routefrom !=
'"+ts+"'";
                                                state2.executeUpdate(sqltext);
                                                //If you changed the direction of the
area, delete any routes that
                                                //need the starting node
                                                while(DList.isEmpty()==false){
                                                        boolean tester3 = false;
                                                        sqltext = "select * from
algorithm where routeto = '"+DList.get(0)+"'";
                                                        rs2 =
state2.executeQuery(sqltext);
                                                        while(rs2.next()){
                                                                tester3 = true;
                                                        }
                                                        if(tester3 == false){
                                                                sqltext = "select
routeto from algorithm where routefrom = '"+DList.get(0)+"' and active = 'y'";
                                                                rs2 =
state2.executeQuery(sqltext);
                                                                while(rs2.next())

        DList.add(rs2.getString(1));
                                                                sqltext = "update
algorithm set active = 'n' where routefrom = '"+DList.get(0)+"'";
                                                        }

                                                        DList.remove(0);
                                                }
                                                //update the table
```

```
                                                        sqltext = "update algorithm set
active = 'y' where routername = '"+tr+"' and routeto = '"+te+"' and routefrom =
'"+ts+"'";
                                                        state2.executeUpdate(sqltext);
                                                        //Update the candidate list
                                                        sqltext = "select distinct routername
from algorithm where routefrom = '"+te+"' or routeto = '"+te+"'";
                                                        rs2 = state2.executeQuery(sqltext);
                                                        while(rs2.next()){

        if(CList.contains(rs2.getString(1))==false)

        CList.add(rs2.getString(1));
                                                        }


                                                }
                                                //This areas that are activated are changed
                                                sareaList.clear();
                                                sareaList.add(areaList.get(i));
                                                sqltext = "select routeto from algorithm
where active = 'y'";
                                                rs2 = state2.executeQuery(sqltext);
                                                while(rs2.next()){

        if(sareaList.contains(rs2.getString(1))==false)

        sareaList.add(rs2.getString(1));
                                                }

                                        }
                                //If done, break out of the loop
                                if(breaker == true)
                                        break;
                                }
                        }

                        //Insert the new routes into output table
                        sqltext = "select routername,routefrom,routeto,ad,active from
algorithm where active = 'y'";
                        rs2 = state2.executeQuery(sqltext);
                        String inter = null;
                        String ipad = null;
                        while(rs2.next()){
                                if(rs2.getString(2).compareTo(areaList.get(i))!=0){

                                        sqltext = "select distinct network(ip) from mainospf
where ospfarea = '"+areaList.get(i)+"' except select distinct network(ip) from mainospf
where ospfarea = '"+areaList.get(i)+"' and routername = '"+rs2.getString(1)+"'";
                                        rs3 = state3.executeQuery(sqltext);
                                        while(rs3.next())
                                                IList.add(rs3.getString(1));
                                        sqltext = "select interface,host(ip) from ritable
where routername = '"+rs2.getString(1)+"' and instance = '"+rs2.getString(2)+"'";
                                        rs3 = state3.executeQuery(sqltext);
                                        if(rs3.next()){
                                                inter = rs3.getString(1);
                                                ipad = rs3.getString(2);
                                        }
                                        sqltext = "select host(ip) from interface where ip
>>= '"+ipad+"' and host(ip) != '"+ipad+"'";
                                        rs3 = state3.executeQuery(sqltext);
                                        if(rs3.next()){
                                                ipad = rs3.getString(1);
                                        }
                                        for(int d=0;d<IList.size();d++){
                                                sqltext = "update output set
ospfredistribute = 'E2',interfaceip = '"+ipad+"',interface = '"+inter+"' where routername
= '"+rs2.getString(1)+"' and network(ip) = '"+IList.get(d)+"'";
```

```
                                        state3.executeUpdate(sqltext);
                                }
                                IList.clear();
                        }
                }
                //reset the algorithm table
                sqltext = "update algorithm set active = 'n'";
                state2.executeUpdate(sqltext);
                sareaList.clear();
                er.clear();
                es.clear();
                ee.clear();
                CList.clear();
        }


}


public void printOutput() throws Exception{
        String sqltext,routerName,temp,temp2,admin;
        ResultSet ro,ro2,ro3;
        List<String> areaList = new ArrayList<String>();
        sqltext = "select distinct(routername) from output order by routername
asc";
        ro = stato.executeQuery(sqltext);
        admin = null;
        //Go through each router
        while(ro.next()){
                routerName = ro.getString(1);
                System.out.println(routerName);
                //Print out information for connected routes
                sqltext = "select text(ip),interface from output where routername =
'"+routerName+"' and typeroute = 'C'";
                ro2 = stato2.executeQuery(sqltext);
                while(ro2.next()){
                        System.out.println("C       "+ro2.getString(1)+" is
directly connected, "+ro2.getString(2));
                }
                //Print out information for static routes
                sqltext = "select text(ip),host(interfaceip) from output where
routername = '"+routerName+"' and typeroute = 'S'";
                ro2 = stato2.executeQuery(sqltext);
                while(ro2.next()){
                        System.out.println("S       "+ro2.getString(1)+" [1/0] via
"+ro2.getString(2));
                }
                //Print out information for OSPF routes
                sqltext = "select
text(ip),host(interfaceip),interface,ospfredistribute from output where routername =
'"+routerName+"' and typeroute = 'O'";
                ro2 = stato2.executeQuery(sqltext);
                while(ro2.next()){
                        int weight = -1;
                        weight =
getWeight(routerName,ro2.getString(1),ro2.getString(2));
                        //Internal routes
                        if(ro2.getString(4).compareTo("NULL") == 0){
                                sqltext = "select intraareaad from ospf where
routername = '"+routerName+"' and ip >>= '"+ro2.getString(2)+"'";
                                ro3 = stato3.executeQuery(sqltext);
                                while(ro3.next()){
                                        admin = ro3.getString(1);
                                }
                                System.out.println("O       "+ro2.getString(1)+"
["+admin+"/"+weight+"] via "+ro2.getString(2)+", 00:00:00, "+ro2.getString(3));
                        }
                        //External routes
```

```
                        else if(ro2.getString(4).compareTo("E2")==0){
                                sqltext = "select externalad from ospf where
routername = '"+routerName+"' and ip >>= '"+ro2.getString(2)+"'";
                                ro3 = state3.executeQuery(sqltext);
                                while(ro3.next()){
                                        admin = ro3.getString(1);
                                }
                                System.out.println("O "+ro2.getString(4)+"
"+ro2.getString(1)+" ["+admin+"/"+weight+"] via "+ro2.getString(2)+", 00:00:00,
"+ro2.getString(3));
                        }
                }
        }
    }

    public void pruner() throws Exception{
        String sqltext,routerName,temp,temp2;
        ResultSet rs,rs2,rs3;
        List<String> areaList = new ArrayList<String>();
        sqltext = "select distinct(routername) from output order by routername
asc";
        rs = state.executeQuery(sqltext);
        while(rs.next()){
                routerName = rs.getString(1);
                sqltext = "select network(ip) from output where routername =
'"+routerName+"' and (typeroute = 'C' or typeroute = 'S')";
                rs2 = state2.executeQuery(sqltext);
                while(rs2.next()){
                        areaList.add(rs2.getString(1));
                }
                sqltext = "select network(ip) from output where routername =
'"+routerName+"' and typeroute = 'O'";
                rs2 = state2.executeQuery(sqltext);
                //Delete OSPF routes that were already connected or static routes
                while(rs2.next()){
                        if(areaList.contains(rs2.getString(1))){
                                sqltext = "delete from output where routername =
'"+routerName+"' and typeroute = 'O' and network(ip) = '"+rs2.getString(1)+"'";
                                state3.executeUpdate(sqltext);
                        }
                }
                areaList.clear();
        }
    }

    public void exex() throws Exception{
        String sqltext,routerName,temp,temp2;
        ResultSet rs,rs2,rs3;
        List<String> areaList = new ArrayList<String>();
        sqltext = "select distinct network(ip) from tempospf";
        rs = state.executeQuery(sqltext);
        while(rs.next())
                areaList.add(rs.getString(1));
        sqltext = "select distinct(routername) from output order by routername
asc";
        rs = state.executeQuery(sqltext);
        while(rs.next()){
                routerName = rs.getString(1);
                sqltext = "select network(ip) from output where routername =
'"+routerName+"' and typeroute = 'O'";
                rs2 = state2.executeQuery(sqltext);
                //Change redistributed connected and static routes to external
                while(rs2.next()){
                        if(areaList.contains(rs2.getString(1))){
                                sqltext = "update output set ospfredistribute = 'E2'
where typeroute = 'O' and routername = '"+routerName+"' and network(ip) =
'"+rs2.getString(1)+"'";
                                state3.executeUpdate(sqltext);
                        }
```

```
                        }
                }
        }

        //Connects to the database
        public void Connector(String ipAddress, String database, String user, String pass)
throws Exception{
                DatabaseMetaData dbmd;
                String url = "jdbc:postgresql://"+ipAddress+"/"+database;
                Properties props = new Properties();
                props.setProperty("user",user);
                props.setProperty("password",pass);
                db = DriverManager.getConnection(url,props);
                dbmd = db.getMetaData();
                state = db.createStatement();
                state2 = db.createStatement();
                state3 = db.createStatement();
                state4 = db.createStatement();
                stato = db.createStatement();
                stato2 = db.createStatement();
                stato3 = db.createStatement();
                statw = db.createStatement();
                statw2 = db.createStatement();
                statw3 = db.createStatement();
        }
}
```

APPENDIX F. EXPERIMENT 1 CONFIGURATION FILES

Router 1

Building configuration...

Current configuration : 1118 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!

```
!
!
interface Loopback0
 ip address 10.10.10.1 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.13 255.255.255.0
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.2.14.14 255.255.255.0
 ip ospf cost 120
 half-duplex
!
router ospf 12
 log-adjacency-changes
 network 10.2.14.0 0.0.0.255 area 2
 distance ospf external 100
!
router ospf 11
 log-adjacency-changes
 network 10.1.13.0 0.0.0.255 area 1
 distance ospf external 200
!
no ip http server
no ip http secure-server
ip classless
!
!
```

```
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

Router 2

```
Current configuration : 1100 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
!
```

```
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.2 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/2
 ip address 10.2.24.24 255.255.255.0
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.23 255.255.255.0
 half-duplex
!
router ospf 22
 log-adjacency-changes
 network 10.2.24.0 0.0.0.255 area 2
 distance ospf external 200
!
router ospf 21
 log-adjacency-changes
```

```
 network 10.1.23.0 0.0.0.255 area 1
 distance ospf external 100
!
no ip http server
no ip http secure-server
ip classless
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

Router 3

```
Current configuration : 1157 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R3
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
```

```
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.3 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 ip address 10.1.35.35 255.255.255.0
 ip ospf cost 310
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.31 255.255.255.0
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.32 255.255.255.0
 half-duplex
```

```
!
router ospf 31
 log-adjacency-changes
 redistribute static metric 399 subnets
 network 10.1.13.0 0.0.0.255 area 1
 network 10.1.23.0 0.0.0.255 area 1
 network 10.1.35.0 0.0.0.255 area 1
!
no ip http server
no ip http secure-server
ip classless
ip route 10.10.10.5 255.255.255.255 10.1.35.53
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

Router 4

```
Current configuration : 1004 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R4
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
```

```
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.4 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
```

```
interface Ethernet1/2
 ip address 10.2.24.42 255.255.255.0
 half-duplex
!
interface Ethernet1/3
 ip address 10.2.14.41 255.255.255.0
 half-duplex
!
router ospf 42
 log-adjacency-changes
 network 10.2.14.0 0.0.0.255 area 2
 network 10.2.24.0 0.0.0.255 area 2
!
no ip http server
no ip http secure-server
ip classless
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

Router 5

```
Current configuration : 955 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R5
!
!
```

```
memory-size iomem 15
ip subnet-zero
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.5 255.255.255.255
!
interface Loopback1
 ip address 10.5.5.2 255.255.255.255
!
interface Ethernet0/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/0
 ip address 10.1.35.53 255.255.255.0
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 no ip address
  shutdown
 half-duplex
!
router ospf 51
 log-adjacency-changes
 redistribute connected metric 290 subnets
 redistribute static metric 300 subnets
 network 10.1.35.0 0.0.0.255 area 1
!
ip classless
```

```
ip route 10.10.10.3 255.255.255.255 10.1.35.35
ip route 192.168.1.0 255.255.255.255 Ethernet1/0
no ip http server
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. EXPERIMENT 2 CONFIGURATION FILES

```
Router 1
Using 958 out of 29688 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R1
!
!
ip subnet-zero
!
!
!
call rsvp-sync
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.1 255.255.255.255
!
interface Ethernet0/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.13 255.255.255.0
 half-duplex
!
interface Ethernet1/2
```

```
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.2.14.14 255.255.255.0
 half-duplex
!
router ospf 12
 log-adjacency-changes
 redistribute ospf 11 subnets
 network 10.2.14.0 0.0.0.255 area 2
 distance ospf external 100
!
router ospf 11
 log-adjacency-changes
 redistribute ospf 12 metric 100 subnets
 network 10.1.13.0 0.0.0.255 area 1
 distance ospf external 200
!
ip classless
no ip http server
!
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
end
```

Router 2

```
Using 947 out of 29688 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R2
!
```

```
!
memory-size iomem 10
ip subnet-zero
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.2 255.255.255.255
!
interface Ethernet0/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 10.3.28.28 255.255.255.0
 shutdown
 half-duplex
!
interface Ethernet1/2
 ip address 10.2.24.24 255.255.255.0
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.23 255.255.255.0
 half-duplex
!
router ospf 22
 log-adjacency-changes
 redistribute ospf 21 subnets
 network 10.2.24.0 0.0.0.255 area 2
 distance ospf external 200
!
router ospf 21
 log-adjacency-changes
 redistribute ospf 22 metric 100 subnets
 network 10.1.23.0 0.0.0.255 area 1
```

```
 distance ospf external 100
!
ip classless
no ip http server
!
!
line con 0
line aux 0
line vty 0 4
!
end
```

Router 3

```
Using 982 out of 29688 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R3
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
ip subnet-zero
ip cef
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.3 255.255.255.255
!
interface Ethernet0/0
 ip address 10.0.35.35 255.255.255.0
 half-duplex
!
interface Ethernet1/0
 ip address 10.1.35.35 255.255.255.0
```

```
 ip ospf cost 310
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.31 255.255.255.0
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.32 255.255.255.0
 half-duplex
!
router ospf 31
 log-adjacency-changes
 redistribute static metric 399 subnets
 network 10.1.13.0 0.0.0.255 area 1
 network 10.1.23.0 0.0.0.255 area 1
 network 10.1.35.0 0.0.0.255 area 1
!
no ip http server
ip classless
ip route 10.10.10.5 255.255.255.255 10.1.35.53
!
!
!
line con 0
line aux 0
line vty 0 4
!
!
end

Router 4
Using 793 out of 29688 bytes
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R4
```

```
!
!
!
!
!
!
memory-size iomem 15
ip subnet-zero
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.4 255.255.255.255
!
interface Ethernet0/0
 no ip address
 shutdown
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Ethernet1/1
 no ip address
 shutdown
!
interface Ethernet1/2
 ip address 10.2.24.42 255.255.255.0
!
interface Ethernet1/3
 ip address 10.2.14.41 255.255.255.0
!
router ospf 42
 log-adjacency-changes
 network 10.2.14.0 0.0.0.255 area 2
 network 10.2.24.0 0.0.0.255 area 2
!
ip classless
no ip http server
!
!
line con 0
```

```
 transport input none
line aux 0
line vty 0 4
!
no scheduler allocate
end

Router 5
Using 1012 out of 29688 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R5
!
!
memory-size iomem 15
ip subnet-zero
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.5 255.255.255.255
!
interface Loopback1
 ip address 10.5.5.2 255.255.255.255
!
interface Loopback2
 no ip address
!
interface Loopback3
 no ip address
!
interface Ethernet1/0
 ip address 10.1.35.53 255.255.255.0
 half-duplex
!
interface Ethernet0/0
 no ip address
 half-duplex
```

```
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 no ip address
 shutdown
 half-duplex
!
router ospf 51
 log-adjacency-changes
 redistribute connected metric 290 subnets
 redistribute static metric 300 subnets
 network 10.1.35.0 0.0.0.255 area 1
!
ip classless
ip route 10.10.10.3 255.255.255.255 10.1.35.35
ip route 192.168.1.0 255.255.255.255 Ethernet1/0
no ip http server
!
!
line con 0
line aux 0
line vty 0 4
!
end
```

APPENDIX H. EXPERIMENT 3 CONFIGURATION FILES

router 1

```
Current configuration : 1168 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
```

```
!
interface Loopback0
 ip address 10.10.10.1 255.255.255.255
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.13 255.255.255.0
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.2.14.14 255.255.255.0
 half-duplex
!
router ospf 12
 log-adjacency-changes
 redistribute ospf 11 subnets
 network 10.2.14.0 0.0.0.255 area 2
 distance ospf external 100
!
router ospf 11
 log-adjacency-changes
 redistribute ospf 12 metric 100 subnets
 network 10.1.13.0 0.0.0.255 area 1
 distance ospf external 200
!
ip http server
no ip http secure-server
ip classless
!
!
```

```
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

router 2

```
Current configuration : 1249 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
```

```
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.2 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/2
 ip address 10.2.24.24 255.255.255.0
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.23 255.255.255.0
 half-duplex
!
router ospf 22
 log-adjacency-changes
 redistribute ospf 21 metric 10 subnets
 network 10.2.24.0 0.0.0.255 area 2
 distance ospf external 200
!
```

```
router ospf 21
 log-adjacency-changes
 redistribute ospf 22 metric 100 subnets
 network 10.1.23.0 0.0.0.255 area 1
 distance ospf external 100
!
ip http server
no ip http secure-server
ip classless
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

router 3

```
Current configuration : 1153 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R3
!
boot-start-marker
boot-end-marker
!
!
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
```

```
!
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.3 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 ip address 10.1.35.35 255.255.255.0
 ip ospf cost 90
 half-duplex
!
interface Ethernet1/1
 ip address 10.1.13.31 255.255.255.0
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 ip address 10.1.23.32 255.255.255.0
```

```
 half-duplex
!
router ospf 31
 log-adjacency-changes
 redistribute static metric 399 subnets
 network 10.1.13.0 0.0.0.255 area 1
 network 10.1.23.0 0.0.0.255 area 1
 network 10.1.35.0 0.0.0.255 area 1
!
ip http server
no ip http secure-server
ip classless
ip route 10.10.10.5 255.255.255.255 10.1.35.53
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

router 4

```
Current configuration : 1001 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R4
!
boot-start-marker
boot-end-marker
!
!
```

```
no network-clock-participate slot 1
no network-clock-participate wic 0
no aaa new-model
ip subnet-zero
ip cef
!
!
!
ip audit po max-events 100
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.4 255.255.255.255
!
interface FastEthernet0/0
 no ip address
  shutdown
 duplex auto
 speed auto
!
interface Ethernet1/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
```

```
!
interface Ethernet1/2
 ip address 10.2.24.42 255.255.255.0
 half-duplex
!
interface Ethernet1/3
 ip address 10.2.14.41 255.255.255.0
 half-duplex
!
router ospf 42
 log-adjacency-changes
 network 10.2.14.0 0.0.0.255 area 2
 network 10.2.24.0 0.0.0.255 area 2
!
ip http server
no ip http secure-server
ip classless
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
!
end
```

router 5

Building configuration...

Current configuration : 952 bytes

```
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname R5
```

```
!
!
memory-size iomem 15
ip subnet-zero
!
!
!
!
!
!
interface Loopback0
 ip address 10.10.10.5 255.255.255.255
!
interface Loopback1
 ip address 10.5.5.2 255.255.255.255
!
interface Ethernet0/0
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/0
 ip address 10.1.35.53 255.255.255.0
 half-duplex
!
interface Ethernet1/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet1/3
 no ip address
  no ip address
 shutdown
 half-duplex
!
router ospf 51
 log-adjacency-changes
 redistribute connected metric 290 subnets
 redistribute static metric 300 subnets
```

```
 network 10.1.35.0 0.0.0.255 area 1
!
ip classless
ip route 10.10.10.3 255.255.255.255 10.1.35.35
ip route 192.168.1.0 255.255.255.255 Ethernet1/0
ip http server
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. EXPERIMENT 1 “SHOW IP ROUTE” FILES

router 1

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C       10.2.14.0/24 is directly connected, Ethernet1/3
C       10.1.13.0/24 is directly connected, Ethernet1/1
O E2    10.5.5.2/32 [200/290] via 10.1.13.31, 00:01:57, Ethernet1/1
O E2    10.10.10.3/32 [200/300] via 10.1.13.31, 00:01:57, Ethernet1/1
C       10.10.10.1/32 is directly connected, Loopback0
O E2    10.10.10.5/32 [200/290] via 10.1.13.31, 00:01:57, Ethernet1/1
O       10.2.24.0/24 [110/130] via 10.2.14.41, 00:02:06, Ethernet1/3
O       10.1.23.0/24 [110/20] via 10.1.13.31, 00:01:58, Ethernet1/1
O       10.1.35.0/24 [110/320] via 10.1.13.31, 00:01:58, Ethernet1/1
     192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [200/300] via 10.1.13.31, 00:01:58, Ethernet1/1
```

router 2

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
O       10.2.14.0/24 [110/20] via 10.2.24.42, 00:02:57, Ethernet1/2
O       10.1.13.0/24 [110/20] via 10.1.23.32, 00:02:58, Ethernet1/3
C       10.10.10.2/32 is directly connected, Loopback0
O E2    10.5.5.2/32 [100/290] via 10.1.23.32, 00:02:58, Ethernet1/3
O E2    10.10.10.3/32 [100/300] via 10.1.23.32, 00:02:58, Ethernet1/3
O E2    10.10.10.5/32 [100/290] via 10.1.23.32, 00:02:58, Ethernet1/3
```

C   10.2.24.0/24 is directly connected, Ethernet1/2
C   10.1.23.0/24 is directly connected, Ethernet1/3
O   10.1.35.0/24 [110/320] via 10.1.23.32, 00:02:59, Ethernet1/3
    192.168.1.0/32 is subnetted, 1 subnets
O E2   192.168.1.0 [100/300] via 10.1.23.32, 00:02:59, Ethernet1/3

router 3
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
C   10.1.13.0/24 is directly connected, Ethernet1/1
O E2   10.5.5.2/32 [110/290] via 10.1.35.53, 00:03:37, Ethernet1/0
C   10.10.10.3/32 is directly connected, Loopback0
S   10.10.10.5/32 [1/0] via 10.1.35.53
C   10.1.23.0/24 is directly connected, Ethernet1/3
C   10.1.35.0/24 is directly connected, Ethernet1/0
    192.168.1.0/32 is subnetted, 1 subnets
O E2   192.168.1.0 [110/300] via 10.1.35.53, 00:03:37, Ethernet1/0

router 4
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C   10.2.14.0/24 is directly connected, Ethernet1/3
C   10.10.10.4/32 is directly connected, Loopback0
C   10.2.24.0/24 is directly connected, Ethernet1/2

router 5
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP

```
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
O       10.1.13.0/24 [110/20] via 10.1.35.35, 00:04:24, Ethernet1/0
C       10.5.5.2/32 is directly connected, Loopback1
S       10.10.10.3/32 [1/0] via 10.1.35.35
C       10.10.10.5/32 is directly connected, Loopback0
O       10.1.23.0/24 [110/20] via 10.1.35.35, 00:04:24, Ethernet1/0
C       10.1.35.0/24 is directly connected, Ethernet1/0
     192.168.1.0/32 is subnetted, 1 subnets
S       192.168.1.0 is directly connected, Ethernet1/0
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. EXPERIMENT 2 "SHOW IP ROUTE" FILES

Router 1
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C 10.2.14.0/24 is directly connected, Ethernet1/3
C 10.1.13.0/24 is directly connected, Ethernet1/1
O E2 10.5.5.2/32 [100/100] via 10.2.14.41, 00:00:12, Ethernet1/3
O E2 10.10.10.3/32 [100/100] via 10.2.14.41, 00:00:12, Ethernet1/3
C 10.10.10.1/32 is directly connected, Loopback0
O E2 10.10.10.5/32 [100/100] via 10.2.14.41, 00:00:12, Ethernet1/3
O 10.2.24.0/24 [110/20] via 10.2.14.41, 00:00:22, Ethernet1/3
O E2 10.1.23.0/24 [100/10] via 10.2.14.41, 00:00:23, Ethernet1/3
O E2 10.1.35.0/24 [100/320] via 10.2.14.41, 00:00:19, Ethernet1/3
192.168.1.0/32 is subnetted, 1 subnets
O E2 192.168.1.0 [100/100] via 10.2.14.41, 00:00:13, Ethernet1/3

Router 2
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
O E2 10.2.14.0/24 [100/100] via 10.1.23.32, 00:01:35, Ethernet1/3
O 10.1.13.0/24 [110/20] via 10.1.23.32, 00:01:35, Ethernet1/3
C 10.10.10.2/32 is directly connected, Loopback0
O E2 10.5.5.2/32 [100/100] via 10.1.23.32, 00:01:35, Ethernet1/3
O E2 10.10.10.3/32 [100/100] via 10.1.23.32, 00:01:35, Ethernet1/3
O E2 10.10.10.5/32 [100/100] via 10.1.23.32, 00:01:35, Ethernet1/3

```
C       10.2.24.0/24 is directly connected, Ethernet1/2
C       10.1.23.0/24 is directly connected, Ethernet1/3
O       10.1.35.0/24 [110/320] via 10.1.23.32, 00:01:36, Ethernet1/3
    192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [100/100] via 10.1.23.32, 00:01:36, Ethernet1/3
```

Router 3

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
O E2    10.2.14.0/24 [110/100] via 10.1.13.13, 00:02:05, Ethernet1/1
C       10.1.13.0/24 is directly connected, Ethernet1/1
O E2    10.5.5.2/32 [110/100] via 10.1.13.13, 00:02:05, Ethernet1/1
C       10.10.10.3/32 is directly connected, Loopback0
S       10.10.10.5/32 [1/0] via 10.1.35.53
O E2    10.2.24.0/24 [110/100] via 10.1.13.13, 00:02:05, Ethernet1/1
                     [110/100] via 10.1.23.23, 00:02:05, Ethernet1/3
C       10.1.23.0/24 is directly connected, Ethernet1/3
C       10.1.35.0/24 is directly connected, Ethernet1/0
    192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [110/100] via 10.1.13.13, 00:02:06, Ethernet1/1
```

Router 4

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C       10.2.14.0/24 is directly connected, Ethernet1/3
O E2    10.1.13.0/24 [110/10] via 10.2.14.14, 00:02:32, Ethernet1/3
O E2    10.5.5.2/32 [110/100] via 10.2.24.24, 00:02:27, Ethernet1/2
```

O E2 10.10.10.3/32 [110/100] via 10.2.24.24, 00:02:27, Ethernet1/2
C 10.10.10.4/32 is directly connected, Loopback0
O E2 10.10.10.5/32 [110/100] via 10.2.24.24, 00:02:27, Ethernet1/2
C 10.2.24.0/24 is directly connected, Ethernet1/2
O E2 10.1.23.0/24 [110/10] via 10.2.24.24, 00:02:33, Ethernet1/2
O E2 10.1.35.0/24 [110/320] via 10.2.24.24, 00:02:33, Ethernet1/2
  192.168.1.0/32 is subnetted, 1 subnets
O E2 192.168.1.0 [110/100] via 10.2.24.24, 00:02:28, Ethernet1/2

Router 5
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
  D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
  N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
  E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
  i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
  ia - IS-IS inter area, * - candidate default, U - per-user static route
  o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

  10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
O E2 10.2.14.0/24 [110/100] via 10.1.35.35, 00:02:56, Ethernet1/0
O 10.1.13.0/24 [110/20] via 10.1.35.35, 00:02:56, Ethernet1/0
C 10.5.5.2/32 is directly connected, Loopback1
S 10.10.10.3/32 [1/0] via 10.1.35.35
C 10.10.10.5/32 is directly connected, Loopback0
O E2 10.2.24.0/24 [110/100] via 10.1.35.35, 00:02:56, Ethernet1/0
O 10.1.23.0/24 [110/20] via 10.1.35.35, 00:02:56, Ethernet1/0
C 10.1.35.0/24 is directly connected, Ethernet1/0
  192.168.1.0/32 is subnetted, 1 subnets
S 192.168.1.0 is directly connected, Ethernet1/0

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K. EXPERIMENT 3 “SHOW IP ROUTE” FILES

router 1

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C       10.2.14.0/24 is directly connected, Ethernet1/3
C       10.1.13.0/24 is directly connected, Ethernet1/1
O E2    10.5.5.2/32 [100/10] via 10.2.14.41, 00:06:20, Ethernet1/3
O E2    10.10.10.3/32 [100/10] via 10.2.14.41, 00:06:20, Ethernet1/3
C       10.10.10.1/32 is directly connected, Loopback0
O E2    10.10.10.5/32 [100/10] via 10.2.14.41, 00:06:20, Ethernet1/3
O       10.2.24.0/24 [110/20] via 10.2.14.41, 00:06:20, Ethernet1/3
O E2    10.1.23.0/24 [100/10] via 10.2.14.41, 00:06:21, Ethernet1/3
O E2    10.1.35.0/24 [100/10] via 10.2.14.41, 00:06:21, Ethernet1/3
     192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [100/10] via 10.2.14.41, 00:06:21, Ethernet1/3
```

router 2

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
O E2    10.2.14.0/24 [100/100] via 10.1.23.32, 00:07:30, Ethernet1/3
O       10.1.13.0/24 [110/20] via 10.1.23.32, 00:07:30, Ethernet1/3
C       10.10.10.2/32 is directly connected, Loopback0
O E2    10.5.5.2/32 [100/100] via 10.1.23.32, 00:07:24, Ethernet1/3
```

```
O E2    10.10.10.3/32 [100/100] via 10.1.23.32, 00:07:24, Ethernet1/3
O E2    10.10.10.5/32 [100/100] via 10.1.23.32, 00:07:24, Ethernet1/3
C       10.2.24.0/24 is directly connected, Ethernet1/2
C       10.1.23.0/24 is directly connected, Ethernet1/3
O       10.1.35.0/24 [110/100] via 10.1.23.32, 00:07:31, Ethernet1/3
    192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [100/100] via 10.1.23.32, 00:07:25, Ethernet1/3
```

router 3

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
O E2    10.2.14.0/24 [110/100] via 10.1.13.13, 00:09:08, Ethernet1/1
C       10.1.13.0/24 is directly connected, Ethernet1/1
O E2    10.5.5.2/32 [110/100] via 10.1.13.13, 00:09:02, Ethernet1/1
C       10.10.10.3/32 is directly connected, Loopback0
S       10.10.10.5/32 [1/0] via 10.1.35.53
O E2    10.2.24.0/24 [110/100] via 10.1.13.13, 00:09:08, Ethernet1/1
                     [110/100] via 10.1.23.23, 00:09:08, Ethernet1/3
C       10.1.23.0/24 is directly connected, Ethernet1/3
C       10.1.35.0/24 is directly connected, Ethernet1/0
    192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [110/100] via 10.1.13.13, 00:09:03, Ethernet1/1
```

router 4

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set
```

```
    10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
C       10.2.14.0/24 is directly connected, Ethernet1/3
O E2    10.1.13.0/24 [110/10] via 10.2.14.14, 00:10:21, Ethernet1/3
                     [110/10] via 10.2.24.24, 00:10:21, Ethernet1/2
O E2    10.5.5.2/32 [110/10] via 10.2.24.24, 00:10:16, Ethernet1/2
O E2    10.10.10.3/32 [110/10] via 10.2.24.24, 00:10:16, Ethernet1/2
C       10.10.10.4/32 is directly connected, Loopback0
O E2    10.10.10.5/32 [110/10] via 10.2.24.24, 00:10:16, Ethernet1/2
C       10.2.24.0/24 is directly connected, Ethernet1/2
O E2    10.1.23.0/24 [110/10] via 10.2.24.24, 00:10:17, Ethernet1/2
O E2    10.1.35.0/24 [110/10] via 10.2.24.24, 00:10:17, Ethernet1/2
    192.168.1.0/32 is subnetted, 1 subnets
O E2    192.168.1.0 [110/10] via 10.2.24.24, 00:10:17, Ethernet1/2
```

router 5

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
O E2    10.2.14.0/24 [110/100] via 10.1.35.35, 00:11:31, Ethernet1/0
O       10.1.13.0/24 [110/20] via 10.1.35.35, 00:11:31, Ethernet1/0
C       10.5.5.2/32 is directly connected, Loopback1
S       10.10.10.3/32 [1/0] via 10.1.35.35
C       10.10.10.5/32 is directly connected, Loopback0
O E2    10.2.24.0/24 [110/100] via 10.1.35.35, 00:11:31, Ethernet1/0
O       10.1.23.0/24 [110/20] via 10.1.35.35, 00:11:31, Ethernet1/0
C       10.1.35.0/24 is directly connected, Ethernet1/0
    192.168.1.0/32 is subnetted, 1 subnets
S       192.168.1.0 is directly connected, Ethernet1/0
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L. HOW TO CREATE THE DATABASE

```
--
-- PostgreSQL database dump
--

SET client_encoding = 'UTF8';
SET check_function_bodies = false;
SET client_min_messages = warning;

--
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres
--

COMMENT ON SCHEMA public IS 'Standard public schema';


SET search_path = public, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: algorithm; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE algorithm (
    routername text,
    routefrom text,
    routeto text,
    ad integer,
    active text
);


ALTER TABLE public.algorithm OWNER TO mcmanst;

--
-- Name: ospf; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE ospf (
    routername text,
```

```
    ospflabel text,
    ip inet,
    ospfarea integer,
    ospfcost integer,
    externalad integer,
    interareaad integer,
    intraareaad integer,
    typeredistribute text
);


ALTER TABLE public.ospf OWNER TO mcmanst;

--
-- Name: instancetab; Type: VIEW; Schema: public; Owner: mcmanst
--

CREATE VIEW instancetab AS
    SELECT ospf.ospfarea AS instanceid, ospf.routername, ospf.externalad AS defaultad
FROM ospf;


ALTER TABLE public.instancetab OWNER TO mcmanst;

--
-- Name: interface; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE interface (
    routername text,
    hostname text,
    interfacename text,
    ip inet
);


ALTER TABLE public.interface OWNER TO mcmanst;

--
-- Name: tempospf; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE tempospf (
    routername text,
    ospflabel text,
```

```
    ip inet,
    ospfarea integer,
    ospfcost integer,
    externalad integer,
    interareaad integer,
    intraareaad integer,
    typeredistribute text
);


ALTER TABLE public.tempospf OWNER TO mcmanst;

--
-- Name: mainospf; Type: VIEW; Schema: public; Owner: mcmanst
--

CREATE VIEW mainospf AS
    SELECT ospf.routername, ospf.ospflabel, ospf.ip, ospf.ospfarea, ospf.ospfcost,
ospf.externalad, ospf.interareaad, ospf.intraareaad, ospf.typeredistribute FROM ospf
UNION SELECT tempospf.routername, tempospf.ospflabel, tempospf.ip,
tempospf.ospfarea, tempospf.ospfcost, tempospf.externalad, tempospf.interareaad,
tempospf.intraareaad, tempospf.typeredistribute FROM tempospf;


ALTER TABLE public.mainospf OWNER TO mcmanst;

--
-- Name: morered; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE morered (
    routername text,
    routefrom text,
    routeto text,
    cost integer
);


ALTER TABLE public.morered OWNER TO mcmanst;

--
-- Name: output; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE output (
```

```
    routername text,
    typeroute text,
    ospfredistribute text,
    ip inet,
    interfaceip inet,
    interface text,
    area text
);


ALTER TABLE public.output OWNER TO mcmanst;

--
-- Name: redistribute; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE redistribute (
    routername text,
    ospfinput text,
    protocol text,
    ospfoutput text,
    cost integer,
    used text
);


ALTER TABLE public.redistribute OWNER TO mcmanst;

--
-- Name: ritable; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--

CREATE TABLE ritable (
    routername text,
    instance text,
    interface text,
    ip inet
);


ALTER TABLE public.ritable OWNER TO mcmanst;

--
-- Name: static; Type: TABLE; Schema: public; Owner: mcmanst; Tablespace:
--
```

```
CREATE TABLE static (
    routername text,
    ip inet,
    interfaceip inet
);


ALTER TABLE public.static OWNER TO mcmanst;

--
-- Data for Name: algorithm; Type: TABLE DATA; Schema: public; Owner: mcmanst
--




--
-- Data for Name: interface; Type: TABLE DATA; Schema: public; Owner: mcmanst
--




--
-- Data for Name: morered; Type: TABLE DATA; Schema: public; Owner: mcmanst
--




--
-- Data for Name: ospf; Type: TABLE DATA; Schema: public; Owner: mcmanst
--




--
-- Data for Name: output; Type: TABLE DATA; Schema: public; Owner: mcmanst
--




--
-- Data for Name: redistribute; Type: TABLE DATA; Schema: public; Owner: mcmanst
--
```

--
-- Data for Name: ritable; Type: TABLE DATA; Schema: public; Owner: mcmanst
--

--
-- Data for Name: static; Type: TABLE DATA; Schema: public; Owner: mcmanst
--

--
-- Data for Name: tempospf; Type: TABLE DATA; Schema: public; Owner: mcmanst
--

--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;

--
-- PostgreSQL database dump complete
--

APPENDIX M. HOW TO CLEAN THE DATABASE

```
delete from algorithm;
delete from interface;
delete from ospf;
delete from output;
delete from redistribute;
delete from ritable;
delete from static;
delete from tempospf;
delete from morered;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX N. HOW TO RUN THE PROGRAM

```
#!/bin/sh
psql -f cleaner
javac *.java
java -classpath /home/mcmanst/code/postgresql-8.1-404.jdbc2.jar:/home/mcmanst/code TestDriver
java -classpath /home/mcmanst/code/postgresql-8.1-404.jdbc2.jar:/home/mcmanst/code FindRoute
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Geoffrey G. Xie. Jibin Zhan, David A. Maltz, Hui Zhang, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. *On Static Reachability Analysis of IP Networks.* IEEE, 2005.

[2] Franck Le, Geoffrey G. Xie, and Hui Zhang. *Understanding Router Redistribution*. 2007.

[3] J. Moy, Internet RFC 2328 *OSPF Version 2*. April 1998. http://www.ietf.org/rfc/rfc2328.txt

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1997

[5] Scott Preston and Preston Research LLC, "Dijkstra.java," 2005, http://www.koders.com/java/fid127BF8AC951DE655E2CD0783CF857936495D5546.aspx.

[6] Scott Preston and Preston Research LLC, "Edge.java," 2005, http://www.koders.com/java/fidFFC47F8AA320EFC5FC01D7B72BA3C03A09803A4B.aspx.

[7] Scott Preston and Preston Research LLC, "Vertex.java," 2005, http://www.koders.com/java/fidCE81790B4B95246C5597C8F2E95B59112702F028.aspx.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Geoffrey Xie
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

4. Professor J.D. Fulp
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California